Foundations and Trends® in Systems and Control
Vol. XX, No. XX (2015) 1–67

DOI: 10.1561/XXXXXXXXXX

now
the essence of knowledge

Operator splitting methods in control

XXX
École Polytechnique Fédérale de Lausanne (EPFL)
XXX

XXX
École Polytechnique Fédérale de Lausanne (EPFL)
XXX

Contents

Abstract

The significant progress that has been made in recent years both in hardware implementations and in numerical computing has rendered real-time optimization-based control a viable option when it comes to advanced industrial applications. More recently, the need for control of a process in the presence of a limited amout of hardware resources has triggered research in the direction of embedded optimization-based control. At the same time, and standing at the other side of the spectrum, the field of big data has emerged, seeking for solutions to problems that classical optimization algorithms are incapable to provide. This triggered some interest to revisit the family of first order methods commonly known as *decomposition schemes* or *operator splitting methods*. Although it is established that splitting methods are quite beneficial when applied to large-scale problems, their potential in solving small to medium scale embedded optimization problems has not been studied so extensively. Our purpose is to study the behavior of such algorithms as solvers of control-related problems of that scale. Our effort focuses on identifying special characteristics of these problems and how they can be exploited by some popular splitting methods.

. *Operator splitting methods in control*. Foundations and Trends® in Systems and Control, vol. XX, no. XX, pp. 1–67, 2015.

DOI: 10.1561/XXXXXXXXXX.

1

Introduction

The significant progress that has been made in recent years both in hardware implementations and in numerical computing has rendered real-time optimization-based control a viable option when it comes to advanced industrial applications. More recently, the need for control of a process in the presence of a limited amout of hardware resources has triggered research in the direction of embedded optimization-based control. Many efficient high-speed solvers have been developed for both linear and nonlinear control, based on either *first order methods* (FiOrdOs [70]), *interior point (IP) methods* (FORCES [22], CVXGEN [49]), *active set methods* (QPOASES [29]) etc.

In this work we focus on systems with linear dynamics, giving rise to convex control problems. The purpose of the survey is to explore a family of first order methods known as *decomposition schemes* or *operator splitting methods*. The abstract form of the problem at hand is the minimization of the sum of two convex functions subject to linear equality constraints, and can be written as

$$\text{minimize} \quad f(x) + g(Ax) \ , \tag{1.1}$$

with variables $x \in \mathcal{X} \subseteq \mathbb{R}^n$, where $f : \mathcal{X} \to (-\infty, \infty]$ and $g : \mathbb{R}^m \to$

$(-\infty, \infty]$ are proper, lower semi-continuous (lsc) convex functions and $A : \mathbb{R}^n \to \mathbb{R}^m$ is a linear map. A splitting method can be applied to the above problem after rewriting it as

$$\begin{array}{ll} \text{minimize} & f(x) + g(z) \\ \text{subject to} & Ax = z \ , \end{array} \tag{1.2}$$

by alternatively (or simultaneously) minimizing over f and g. A dual variable update for the equality constraint ensures that the solutions of problems (1.2) and (1.1) are identical. Inequality constraints that might appear are already embedded in one of the two functions in the form of indicator functions, *i.e.*, a membership function for a set $\mathcal{C}$

$$\delta_{\mathcal{C}}(x) = \begin{cases} 0 & x \in \mathcal{C} \\ \infty & \text{otherwise.} \end{cases} \tag{1.3}$$

Formulations similar to the above have been studied extensively and we can look for their roots in the method of multipliers [45], [61], the Arrow-Hurwicz method [46], Douglas-Rachford splitting [23] and ADMM [34], [30]. Other classical references include [9] and [24]. More recent references that illustrate the applicability of such methods in modern engineering problems (signal and image processing, big data analysis, machine learning) are [13] and [17]. The thesis [26] provides a comprehensive description of the connection of several splitting algorithms under a common framework. Finally, the book [3] provides a mathematically rigorous introduction to operator splitting methods in general Hilbert spaces.

The plethora of different approaches for solving problem (1.2) is partly a consequence of the problem-dependent behavior of first-order methods. This behavior has both its pros and cons; on one hand, sensitivity to the problem's structure and data requires preprocessing and tuning of several parametes, a procedure that can be cumbersome. However, it is exactly this procedure that gives the flexibility to customize the solver to the problem at hand, and, in many cases, beat by several orders of magnitude general purpose solvers. Consequently, there are numerous approaches, each of which can be less or more pertinent for the specific problem. Mentioning some of the most important

categorizations, we can solve either the *primal* (primal decomposition [9]) or the *dual* problem by means of *Lagrangian relaxation* (dual decomposition [20], [27], [68], [8]), *augmented Lagrangian relaxation* [7], [64], [65], *alternating minimization (Gauss-Seidel) augmented Lagrangian schemes* (ADMM), mixture of Lagrangian with augmented Lagrangian schemes (AMA [69]), *linearized augmented Lagrangians* or *approximate minimization* schemes (L-PMM [15], PADMM [1]), *mixtures of alternating minimization with partial linearization* (primal-dual hybrid gradient [48], [25], Chambolle-Pock algorithm [14], [19]).

Although it is well-established that splitting methods are quite beneficial when applied to large-scale problems, their potential in solving small to medium scale embedded optimization problems has not been studied in so extensively. Our purpose is to study the behavior of such algorithms as solvers of control-related problems of that scale. Our effort focuses on identifying special characteristics of these problems and how they can be exploited from some popular splitting methods. Some of the questions that we attempt to answer are:

1. It is very common in practice that optimal control problems come with a quadratic objective, since in this way stability can be proven for regulation or tracking purposes. What is the best way to exploit this smooth term?

2. Given that a control problem has to be solved repeatedly (*e.g.*, MPC), how does warm-starting of the solution affect the speed?

3. Given the structure of the problem at hand, which algorithms will converge more quickly?

4. Are there ways to precondition the problem in order to reduce the solve time?

In what follows we present three well-understood splitting algorithms, the *Alternating direction method of multipliers (ADMM)*, the *Alternating minimization algorithm (AMA)* and the *primal-dual hybrid gradient scheme (PDHG)*. These three methods come from different sides of the spectrum described above, but also hold very

strong similarities. Our choice is motivated from the fact that the methods are analyzed and extended from several communities, and hence their properties are well-understood.

The paper is organized as follows: In Section 2 we formulate the problem we want to solve and look at it from three different perspectives, resulting to the three algorithms we use. Subsequently we introduce the algorithms under a unified scheme and report their properties. In Section XXX we discuss how they can be applied to our problem formulation. In the next two sections we get past the basic variants of the methods presented before, we introduce several enhanced versions and we focus on their applicability to solving optimization problems. More specifically, in Section **??** we explain how one can exploit the structure of the problem to accelerate the theoretical convergence rates. In Section XXX we discuss the computational aspects; we propose existing methods that can speedup a linear system solve that might occur, as well as a few offline preconditioning techniques. Finally, the algorithms are illustrated with XXX examples in Section 8.

2

The algorithms

We narrow the general formulation to our problems of interest which can, without loss of generality, be written as

$$\begin{array}{ll} \text{minimize} & (1/2)z^T Qz + c^T z + \sum_{i=1}^{M} g_i(T_i z + t_i) \\ \text{subject to} & Az = b \ , \end{array} \tag{P}$$

with variable $z \in \mathcal{Z} \subseteq \mathbb{R}^n$, and $\mathcal{Z}$ is a convex set. It holds that $Q \in \mathbb{S}^n_+$, $T_i \in \mathbb{R}^{p_i \times n}$ normally have a sparse and banded structure. Equality constraints are stated by means of the matrix $A \in \mathbb{R}^{m \times n}$ and the vector b. The following assumption holds:

Assumption 1. The functions $g_i : \mathbb{R}^{p_i} \to (-\infty, \infty]$ are closed, lsc convex functions.

Formulation (P) is quite general and can describe any convex optimization problem. The choice of the quadratic part $(1/2)z^T Qz + c^T z$ and the equality constraints $Az = b$ being represented in an explicit way is motivated by the standard form of control problems. The set $\mathcal{Z}$ would typically be $\mathcal{Z} = \mathbb{R}^n$, since the constraints are usually expressed through indicator functions g_i, although it is sometimes more efficient to embed simple constraints directly in $\mathcal{Z}$. In what follows we make use

of the default assumption $\mathcal{Z} = \mathbb{R}^n$, unless explicitely stated otherwise.

It is important to mention that the original formulation (1.2) involves two functions in the objective, while in (P) we consider *two groups of functions*. The first group contains only $f(z) := \Big\{(1/2)z^T Q z + c^T z + \delta_{\mathcal{D}}(z)\Big\}$, $f(z) : \mathcal{Z} \to \mathbb{R}$, the quadratic objective restricted to the subspace spanned by the dynamics equation. Note that we used the indicator function

$$\delta_{\mathcal{D}}(z) = \begin{cases} 0 & Az = b \\ \infty & \text{otherwise.} \end{cases}$$

The second group constitutes of M functions $g_i(y_i)$. By concatenating the vectors and matrices associated with the affine term in g_i's as $T = (T_1, \ldots, T_M)$ and $t = (t_1, \ldots, t_M)$, and by introducing slack variables $y_i = T_i z + t_i,\ i = 1, \ldots, M$, we can recast (P) as

$$\begin{array}{ll} \text{minimize} & f(x) + g(y) \\ \text{subject to} & Tz - y = -t \ , \end{array}$$

where $g(y) = \sum_{i=1}^{M} g_i(y_i)$, $g(y) : \mathcal{Y} \to (-\infty, \infty]$ and $\mathcal{Y} \subseteq \mathbb{R}^p$, $p = \sum_{i=1}^{M} p_i$. Thus we end up with the original formulation; nonetheless, this seemingly innocent recasting can significantly affect the implementation of the algorithms, as we will see in the next sections.

The algorithms of interest solve problem (P) by considering different formulations. Next, we present all the equivalent forms of (P) that are going to be used in the rest of the article.

2.1 Equivalent reformulations of the problem

We start by deriving the *Lagrangian* for (P), which can be written as

$$\mathcal{L} = f(z) + \sum_{i=1}^{M} g_i(y_i) + \sum_{i=1}^{M} \langle \lambda_i, T_i z + t_i - y_i \rangle \ , \qquad \text{(L)}$$

where $\lambda_i \in \mathbb{R}^{p_i}$ are dual variables associated with the equality constraints introduced above. The optimum can be recovered by solving

$$(\lambda^\star, z^\star, y^\star) = \operatorname*{argmax}_{\lambda} \operatorname*{argmin}_{z,y} \mathcal{L}(\lambda, z, y) \ , \tag{SolveL}$$

where $\lambda = (\lambda_1, \ldots, \lambda_M) \in \mathbb{R}^p$.

One can also derive the dual of (P) by means of the Lagrangian formulation. We have that

$$\begin{aligned} d(\lambda) &= \min_{z,y} \left\{ f(z) + \sum_{i=1}^{M} g_i(y_i) + \sum_{i=1}^{M} \langle \lambda_i, T_i z + t_i - y_i \rangle \right\} \\ &= -\max_{z} \left\{ -f(z) - \left\langle T^T\lambda, z \right\rangle \right\} - \max_{y} \left\{ -\sum_{i=1}^{M} g_i(y_i) + \langle y, \lambda \rangle \right\} + \langle t, \lambda \rangle \, . \end{aligned}$$

Making use of the Legendre-Fenchel conjugate (see, *e.g.*, [12, Chapter 3]), the dual function can be expressed as

$$d(\lambda) = -f^\star(-T^T\lambda) - \sum_{i=1}^{M} g_i^\star(\lambda_i) + \langle t, \lambda \rangle \ \ , \tag{D}$$

and the dual problem to solve becomes

$$\lambda^\star = \operatorname*{argmin}_{\lambda} \ f^\star(-T^T\lambda) + \sum_{i=1}^{M} g_i^\star(\lambda_i) - \langle t, \lambda \rangle \ \ . \tag{SolveD}$$

In the algorithms considered here, both formulations (P) and (D) play a significant role. Although we mostly deal with the primal problem (P), the dual counterpart is essential for drawing a complete picture of the relations between the several methods. We consider three approaches, namely solving a saddle point problem either on the Lagrangian, the augmented Lagrangian function or a generic saddle-point formulation that involves the Legendre-Fenchel duals of the functions $g_i(\cdot)$. In order for any of these solutions to be valid, we make the following standing assumption:

Assumption 2. The Lagrangian (L) associated to (P) has a saddle point, *i.e.*,

$$\mathcal{L}(\lambda, z^\star, y^\star) \leq \mathcal{L}(\lambda^\star, z^\star, y^\star) \leq \mathcal{L}(\lambda^\star, z, y) \quad \forall \lambda, z, y \in \mathbb{R}^p \times \mathcal{Z} \times \mathbb{R}^p \ \ .$$

An alternative solution to (P) is to consider the *augmented Lagrangian* formulation, defined by

$$\mathcal{L}_\rho = f(z) + \sum_{i=1}^{M} g_i(y_i) + \sum_{i=1}^{M} \langle \lambda_i, T_i z + t_i - y_i \rangle + \frac{\rho}{2} \sum_{i=1}^{M} \|T_i z + t_i - y_i\|^2 \ , \tag{AL}$$

for $\rho > 0$ and the associated problem to solve becomes

$$(\lambda^\star, z^\star, y^\star) = \underset{\lambda}{\operatorname{argmax}} \underset{z,y}{\operatorname{argmin}} \ \mathcal{L}_\rho(\lambda, z, y) \ . \tag{SolveAL}$$

Another option is to apply some partial dualization to the initial formulation (P), resulting in a primal-dual equivalent that is easier to solve. Making use of the Legendre-Fenchel conjugate the functions $g_i(T_i z + t_i)$ and $\delta_{\mathcal{D}}(z)$ can be expressed as

$$\begin{aligned} g_i(T_i z + t_i) &= \sup_{\lambda_i} \left\{ \langle T_i z + t_i, \lambda_i \rangle - g_i^\star(\lambda_i) \right\} \ , \\ \delta_{\mathcal{D}}(z) &= \sup_{y} \left\{ \langle y, z \rangle - S_{\mathcal{D}}(y) \right\} \ , \end{aligned}$$

where $S_{\mathcal{D}}(y)$ denotes the *support function for the dynamics set*. Consequently we can solve

$$\begin{aligned} (z^\star, \lambda^\star, y^\star) = \underset{z}{\operatorname{argmin}} \underset{y,\lambda}{\operatorname{argmax}} \Big\{ & \sum_{i=1}^{M} \langle T_i z + t_i, \lambda_i \rangle + \langle z, y \rangle + \frac{1}{2} z^T Q z \\ & + c^T z - \sum_{i=1}^{M} g_i^\star(\lambda_i) - S_{\mathcal{D}}(y) \Big\} \ . \end{aligned} \tag{PD}$$

For indicator functions of convex cones, the Legendre-Fenchel dual is the indicator function of the polar cone, rendering the evaluation of $g_i^\star$ easy for the standard self-dual cones. The saddle-point formulation in (PD) can be written more compactly by defining the function

$$\mathcal{S} = \sum_{i=1}^{M} \langle T_i z + t_i, \lambda_i \rangle + \langle z, y \rangle + \frac{1}{2} z^T Q z + c^T z - \sum_{i=1}^{M} g_i^\star(\lambda_i) - S_{\mathcal{D}}(y) \ . \tag{S}$$

Hence, we have that

$$(z^\star, \lambda^\star, \nu^\star) = \underset{z}{\operatorname{argmin}} \underset{\lambda,\nu}{\operatorname{argmax}} \ \mathcal{S}(z, \lambda, \nu) \ . \tag{SolveS}$$

Remark 2.1. In the more general case we are interested in tackling problems of the form

$$\begin{array}{ll} \text{minimize} & (1/2)z^TQz + c^Tz + \sum_{i=1}^{M} \alpha_i g_i(T_i z + t_i) \\ \text{subject to} & Az = b \ , \end{array}$$

where $\alpha_i > 0$ are weights on the functions $g_i(\cdot)$ (think, *e.g.*, of weighted norms). The Lagrangian and augmented Lagrangian functions change trivially. The conjugate of g_i becomes

$$g_i(T_i z + t_i) = \sup_{\lambda_i} \langle T_i z + t_i, \lambda_i \rangle - \alpha_i g_i^\star(\lambda_i/\alpha_i) \ ,$$

where we used standard properties of the conjugate function, given in Appendix A. Accordingly, the saddle point problem to solve becomes

$$\mathcal{S} = \langle Tz + t, \lambda \rangle + \langle z, y \rangle + \frac{1}{2} z^T Q z + c^T z - \sum_{i=1}^{M} \alpha_i g_i^\star(\lambda_i/\alpha_i) - S_{\mathcal{D}}(y) \ .$$

Although the changes follow easily, it might not be the case that the corresponding optimization problems are simple to solve as we will see in the subsequent chapters.

2.2 Origin of the methods and a unified framework

The three approaches for solving (P), *i.e.*, (SolveL),(SolveAL) and (SolveS) originate from Rockafellar's *Proximal method of multipliers* [65] and, via decomposition, a unified framework for the three algorithms can be obtained. We follow the arguing from the recent manuscript [67], which clarifies several aspects of popular splitting methods and shows their close connection.

For the purpose of analyzing the three schemes, we introduce the *proximal augmented Lagrangian* for (P), defined by

$$\begin{aligned} \mathcal{P}_\rho = & f(z) + \sum_{i=1}^{M} g_i(y_i) + \sum_{i=1}^{M} \langle \lambda_i, T_i z + t_i - y_i \rangle + \\ & \frac{\rho}{2} \sum_{i=1}^{M} \|T_i z + t_i - y_i\|^2 + \frac{1}{2} \|z - z^k\|_{P_1}^2 + \frac{1}{2} \sum_{i=1}^{M} \|y_i - y_i^k\|_{P_{2i}}^2 \ , \end{aligned} \tag{PAL}$$

where $P_1 \in \mathbb{S}^n_+$ and $P_{2i} \in \mathbb{S}^{p_i}_+$.

Applying the multiplier method to (PAL), one gets the Proximal method of multipliers (PMM) algorithm:

Algorithm 1 Proximal Method of Multipliers

Require: Initialize $z^0 \in \mathbb{R}^n$, $y^0 \in \mathbb{R}^p$, $\lambda^0 \in \mathbb{R}^p$, and $\rho > 0$

loop

1: $(z^{k+1}, y^{k+1}) \in \underset{z,y}{\operatorname{argmin}}\ \mathcal{P}_\rho$

2: $\lambda^{k+1} = \lambda^k + \rho(Tz^{k+1} + t - y^{k+1})$.

end loop

This is the most general form of the multiplier method. One observes that if (L) is considered instead of (PAL), the *dual decomposition* algorithm is recovered, while if one minimizes (AL) we recover the classical *method of multipliers*.

Step 1 of Algorithm 1 is not necessarily easy to compute and splitting method schemes address exactly this problem, *i.e.*, the decomposition of this minimization problem to two, one that involves the variables z and one that involves y. Three main strategies for achieving this are listed in [67]:

- Linearization with respect to both variables (z, y) (approximate minimization).
- Alternating (Gauss-Seidel) minimization.
- Mixture of the two, *i.e.*, partial linearization with respect to one variable and alternating minimization.

All methods we are going to discuss here are based on one of the above strategies. We introduce the general splitting scheme, the *Proximal alternating direction method of multipliers (PADMM)*, which is written as:

Algorithm 2 Proximal Alternating Direction Method of Multipliers (PADMM)

Require: Initialize $z^0 \in \mathbb{R}^n$, $y_i^0 \in \mathbb{R}^{p_i}$, $\lambda^0 \in \mathbb{R}^{p_i}$, and $\rho > 0$

loop

1: $$z^{k+1} = \underset{z}{\operatorname{argmin}} \quad f(z) + \sum_{i=1}^{M} \left\langle \lambda_i^k, T_i z \right\rangle + (\rho/2) \sum_{i=1}^{M} \|T_i z + t_i - y_i^k\|^2 + (1/2)\|z - z^k\|_{P_1}^2$$

2: $$y_i^{k+1} = \underset{y_i}{\operatorname{argmin}} \quad g_i(y_i) - \left\langle \lambda_i^k, y_i \right\rangle + (\rho/2)\|T_i z^{k+1} + t_i - y_i\|^2 + (1/2)\|y_i - y_i^k\|_{P_{2i}}^2, \ i = 1, \ldots, M$$

3: $$\lambda_i^{k+1} = \lambda_i^k + \rho(T_i z^{k+1} + t_i - y_i^{k+1}), \ i = 1, \ldots, M$$

end loop

Algorithm 2 comes with many names, *e.g.*, *Linearized proximal method of multipliers (L-PMM)* in [67], *Split Inexact Uzawa (SIU)* in [72], *Generalized Alternating Direction Method of Multipliers (GADMM)* in [21].

The existence of the proximal regularizers in light red offers some additional flexibility, which comes by choosing the matrices P_1 and P_2 as desired. Furthermore, they ensure strong convexity of both optimization objectives, and hence uniqueness of the corresponding minimizers.

The second step of the algorithm is a *proximal minimization step* and can be written via the *prox* operator of a function, defined as

$$\mathbf{prox}_{\rho f}(x) := \inf_{y \in Y} \left\{ f(y) + \frac{1}{2\rho}\|y - x\|^2 \right\} \ .$$

A big number of proximal operators have a closed form representation [59], a characteristic that is highly desirable when we apply operator splitting algorithms.

From Algorithm 2 we can recover:

- **Alternating direction method of multiplier (ADMM) [34], [30], [35]**: This is probably the most popular of the splitting methods, mostly due to its simplicity and the very few assumptions for convergence in comparison to other splitting schemes. It

is also known as *split Bregman method* [39]. In order to recover it from Algorithm 2, we set $P_1 = 0$ and $P_{2i} = 0$.

Algorithm 3 Alternating direction method of multiplier (ADMM)

Require: Initialize $z^0 \in \mathbb{R}^p$, $\lambda^0 \in \mathbb{R}^p$, and $\rho > 0$

loop

1: $z^{k+1} = \underset{z}{\operatorname{argmin}} \quad f(z) + \sum_{i=1}^{M} \left\langle \lambda_i^k, T_i z \right\rangle + (\rho/2) \sum_{i=1}^{M} \|T_i z + t_i - y_i^k\|^2$

2: $y_i^{k+1} = \mathbf{prox}_{\frac{1}{\rho} g_i} \left(T_i z^{k+1} + t_i + \lambda_i^k / \rho \right), \ i = 1, \ldots, M$

3: $\lambda_i^{k+1} = \lambda_i^k + \rho(T_i z^{k+1} + t_i - y_i^{k+1}), \ i = 1, \ldots, M$

end loop

It is important to mention that ADMM can be derived from the *Douglas-Rachford splitting method* when the latter is applied to the dual problem (D) (see [24] for more details). We are not going to present the algorithm here, but it will be a useful future reference since many of the convergence results for ADMM have been indirectly derived through the convergence properties of the Douglas-Rachford algorithm.

- **Alternating minimization algorithm (AMA) [69]**: The algorithm is a hybrid scheme, consisting of minimizing the original Lagrangian (L) in Step 1, and the augmented one (AL) in Step 2. In this way, the quadratic coupling that comes from the augmented Lagrangian term vanishes, allowing for further decomposition if the structure of f permits to do so. Matrices P_1, P_{2i} are set as in ADMM. In order to ensure convergence, the stepsize ρ has to be taken as $\epsilon \leq \rho \leq \frac{2\sigma_f}{\|T\|^2} - \epsilon$, where $\epsilon \in (0, \frac{\sigma_f}{\|T\|^2})$ and f has to be strongly convex, with convexity modulus σ_f.

Algorithm 4 Alternating minimization algorithm (AMA)

Require: Initialize $\lambda^0 \in \mathbb{R}^p$, and $\epsilon \leq \rho \leq \frac{2\sigma_f}{\|T\|^2} - \epsilon$, where $\epsilon \in (0, \frac{\sigma_f}{\|T\|^2})$

loop

1: $z^{k+1} = \underset{z}{\operatorname{argmin}} \quad f(z) + \sum_{i=1}^{M} \left\langle \lambda_i^k, T_i z \right\rangle$

2: $y_i^{k+1} = \mathbf{prox}_{\frac{1}{\rho} g_i} \left(T_i z^{k+1} + t_i + \lambda_i^k / \rho \right), \; i = 1, \ldots, M$

3: $\lambda_i^{k+1} = \lambda_i^k + \rho (T_i z^{k+1} + t_i - y_i^{k+1}), \; i = 1, \ldots, M$

end loop

AMA is equivalent to the proximal gradient method applied to the dual problem (D). The derivation of this result is presented in Appendix C.1.

- **Primal-Dual Algorithm (PDA)**: The saddle point formulation (S) has inspired the development of numerous algorithms, extending beyond the original formulation where the objective is the sum of two functions, to multiple ones. The key assumption is that the functions have easy-to-compute proximal operators. In the case where only two functions are involved, the most popular scheme is *Chambolle and Pock's method* [14]. It was originally introduced in [75] and further analyzed in [14] and [42]. It solves problem (SolveS) by means of an alternating scheme which is equivalent to Algorithm 2 for the special choices $P_{2i} = 0$ and $P_1 = (1/\tau) I - \rho \sum_{i=1}^{M} T_i^T T_i$ (see [67]). In this way, Chambolle and Pock's method linearizes the quadratic term that appears in Step 1 of Algorithm 2 and hence decouples the minimization problem.

  Moving to the case of more than two separable functions, generalizations of the algorithmic scheme in [14] appeared in [18] and [71]. Since the aforementioned primal-dual schemes are very similar and come under different names, we will refer to this *family of algorithms* from now on as *Primal-Dual Algorithms (PDA)*. The basic version is presented below.

Algorithm 5 Primal-Dual Algorithm (PDA)

Require: Initialize $\lambda^0 \in \mathbb{R}^p$, $z^0 \in \mathbb{R}^n, y^0 \in \mathbb{R}^m$.

Choose stepsizes $\tau, \rho_1, \ldots, \rho_{M+1} > 0$ such that $2\min\{\frac{1}{\tau}, \frac{1}{\rho_1}, \ldots, \frac{1}{\rho_{M+1}}\}\left(1 - \sqrt{\tau\left(\sum_{i=1}^{M}\rho_i\|T_i\|^2 + \rho_{M+1}\|A\|^2\right)}\right) > 1$

loop

1: $z^{k+1} = \underset{z\in Z}{\operatorname{argmin}} \quad (1/2)z^TQz + c^Tz + \sum_{i=1}^{M}\left\langle z, T_i^T\lambda_i^k\right\rangle + \left\langle z, y^k\right\rangle + (1/2\tau)\|z - z^k\|^2$

2: $\hat{z}^{k+1} = 2z^{k+1} - z^k$

3: $\lambda_i^{k+1} = \mathbf{prox}_{\rho g_i^\star}\left(\lambda_i^k + \rho_i(T_i\hat{z}^{k+1} + t_i)\right),\ i = 1, \ldots, M$

4: $y^{k+1} = \mathbf{prox}_{\rho S_{\mathcal{D}}}\left(y^k + \rho_{M+1}\hat{z}^{k+1}\right)$

end loop

The algorithm amounts to a sequence of proximal minimization steps, all decoupled from one another. Each step admits a closed form solution. An obvious advantage is that, if the matrix Q is diagonal, Step 1 ends up being very cheap, in constrast to ADMM and AMA. The dynamics are also treated separately at Step 4 (see Appendix A.1 for the derivation). The cost of simplifying the optimization problem comes, as in AMA, with restrictions on the stepsizes τ and ρ_i, as the condition suggests.

2.3 Termination

Given that problem (P) is convex, necessary and sufficient conditions for convergence are the *primal* and *dual feasibility conditions* (see, *e.g.*, [12, Chapter 5]). Writing the conditions for formulations (L) and (S), we get termination criteria for ADMM, AMA and PDA, respectively. The optimality conditions practically translate to primal and dual residuals that (asymptotically) go to zero as the algorithmic schemes progress. In practice, the algorithms are terminated when the conditions are satisfied to some prespecified accuracy. In Appendix B we present in detail how the derivation is performed for each of the algorithms.

We should note that, traditionally, first order (splitting) methods

do not provide information about the feasibility of the problem. In the case of infeasibility, the subproblems to which the original problem is split will not converge to a common solution, something that is reflected in the residuals that do not decrease. After having observed this behavior for some time, the user can terminate the algorithm and claim infeasibility. It is only recently that the authors of [58] suggested an ADMM scheme that also returns a feasibility certificate. The idea is to use *homogeneous self-dual embedding*, a method commonly used with interior-point methods [74], [73], and initially used for solving LPs [36]. The original problem is written as a feasibility problem by embedding the KKT conditions into a system of linear equations. From the solution of the embedding problem, one can either recover the solution to the original one, or a certificate for primal or dual infeasibility. In this manuscript we do not consider these variants, which should be used if infeasibility detection is crucial for the problem at hand.

3

How to split

The first question that comes to mind when making use of a splitting method is how to perform the splitting. This choice can heavily affect the speed of the algorithm. Choosing a splitting pattern is equivalent to formulating the two subproblems that have to be solved in any of the algorithmic schemes 3, 4 or 5. Consequently, the choice will also confine the options for acceleration. A general guideline would be the following:

1. Both subproblems should have a closed form solution if possible; if not, they should be cheap to solve. The whole purpose of using splitting on (P) is to end up with simpler subproblems.

2. More precisely, the proximal steps should be simple to solve. The step constitutes often of projections onto simple constraint sets, or proximal minimizations with respect to norms.

3. Expensive operations, like matrix inversions, should be avoided as much as possible. If there are quantities that do not change during the execution, they should be prefactored.

4. If an accelerated version of an algorithm can be used without

heavily altering a well-structured problem, then it should be used.

Below, we examine each method separately, in more detail.

ADMM In this case, most of the flexibility comes in Step 2, since Step 3 is either a simple projection or a proximal minimization operation, provided $g_i(\cdot)$ is simple. The augmented Lagrangian term will contribute with a term of the form $(\rho/2)z^T \sum_{i=1}^{M} T_i^T T_i z$ to the objective, hence even if Q is a diagonal matrix, the resulting quadratic term is most probably dense. In this sense, one can either minimize the resulting quadratic function restricted to the subspace $Az = b$, which will result to the solution of a KKT system (see [57]), or by eliminating the equality constraint. Note that this is equivalent to taking a Newton step on a quadratic perturbation of $f(z)$, which explains why this approach works quite well when it comes to number of iterations for convergence. The bottleneck is the matrix inversion that has to be performed at each iteration. If ρ is constant, one can use either a sparse LDL factorization on the KKT system, or a Cholesky factorization in the second case and consequently solve by means of forward-backward substitution [12, Appendix C].

AMA The method is applicable with the (sometimes limiting) assumption that f is strongly convex (assuming $g_i,\ i = 1, \ldots, M$ are strongly convex would also suffice, but would be more improbable). On the other hand, if the assumption holds and f has some structure (*e.g.*, diagonal, block diagonal), the method should be preferred since the matrix inversion can be very cheap. In several MPC applications this is not the case though, since, in order to ensure strong convexity, f becomes a dense quadratic form for the condensed problem. Note that the spectral radius of T and the minimum eigenvalue of the quadratic term will affect the choice of the stepsize, many times leading to a very small one.

PDA This method combines properties of the other two, in the sense that the first step is still decoupled but there is no strong convexity assumption. In order to avoid densification of the quadratic term, we choose to treat the equality constraints separately (Step 4), a choice that, along with the stepsizes' limitations, can render the algorithm slow to converge in iterations' number. Keeping Step 1 simple allows for moving some (simple) constraints directly in the objective, if the resulting optimization problem has a closed form solution. The algorithm is built such that it favors simple computations in the expense of more iterations; a characteristic that was exploited in the recent manuscript [47].

There are various extensions of the three methods we presented that can significantly improve their performance in practical applications. In general there are two ways to improve timings:

1. Improving the theoretical convergence rates, which is done by exploiting properties of the functions in (P).

2. Practical speedup either on the computations or the iterations count. This can be done in several ways, *e.g.*, exploit structure of the problem, fast numerical linear algebra, preconditioning of the data.

In many cases the approaches described above are competing. For example, one can precondition the problem so that an accelerated variant of a method can be used, but at the same time some favorable sparsity pattern of the original problem is lost. In our experience, there is no 'golden rule' when it comes to choosing a particular method and applying the various extensions for speeding it up. The choice of the method should be motivated from the problem's structure and vice-versa. In the subsections that follow we aim at providing the reader with a wide overview of several variants of the methods that improve the convergence rates, as well as some ideas for efficient linear system solve and preconditioning of the data. In the last section of the article we demonstrate by means of examples how we would make use of these options.

4

Convergence results and accelerated variants

Slow convergence is usually the case with first-order methods, and the algorithms presented above are no exception to this rule. The slow rate can be caused both by the primal and dual iterates. Indeed, the proximal step usually amounts to a projected gradient iteration, while the dual update is a gradient update step. In this aspect, the algorithms presented here cannot achieve a rate better than the existing upper bounds on first-order methods [52], [54].

In what follows we frequently refer to convergence *in function values* and *in sequence values*. By saying that 'we have $O(1/k)$ global rate of convergence in *function values* for some function f, we mean that

$$f(x^k) - p^\star \leq \frac{M}{k} \ ,$$

where $p^\star$ is the optimal value of f and $M > 0$. Accordingly, 'global $O(1/k)$ convergence rate *of a sequence* $\{x^k\}$' means that

$$\|x^k - x^\star\| \leq \frac{M}{k} \ ,$$

where $x^\star$ is the optimizer and $M > 0$. Finally, 'ergodic convergence rate of $O(1/N)$ for a sequence $\{x^k\}$' (or in the function values of f)

means that, for $\bar{x}^N = \frac{1}{N}\sum_{k=1}^N x^k$,

$$\|\bar{x}^N - x^\star\| \leq \frac{M}{N}, \quad f(\bar{x}^N) - p^\star \leq \frac{M}{N} \ ,$$

respectively.

The convergence results for the splitting methods we examine basically derive from existing convergence results of algorithms as simple as the (proximal) gradient method. Common sense dictates that the more knowledge we have about the functions at hand, the better the convergence rates we can derive. Consequently, structural assumptions are crucial for converging faster to optimality. The most important ones are:

Definition 4.1. (Smoothness): A closed convex lower semicontinous function $f : \mathcal{Z} \to \mathbb{R}$ is β-smooth if it is differentiable and

$$f(z) \leq f(x) + \langle \nabla f(x), z - x \rangle + \frac{\beta}{2}\|z - x\|^2 \ ,$$

holds for all $z, x \in \mathcal{Z}$.

Definition 4.2. (Strong convexity): A closed convex lower semicontinous function $f : \mathcal{Z} \to \mathbb{R}$ is β-strongly convex if

$$f(z) \geq f(x) + \langle u, z - x \rangle + \frac{\beta}{2}\|z - x\|^2 \ ,$$

holds for all $z, x \in \mathcal{Z}$ and all $u \in \partial f(x)$.

Remark 4.1. The above definitions generalize in the case of positive semi-definite matrices M instead of scalars β. The inequalities then hold with $\frac{1}{2}\|z - x\|_M^2$.

Definition 4.3. (Lipschitz continuity): A closed convex lower semicontinous and differentiable function $f : \mathcal{Z} \to \mathbb{R}$ is β-Lipschitz continuous if

$$\|\nabla f(z) - \nabla f(x)\| \leq \beta\|z - x\| \ .$$

β-Lipschitz continuity is equivalent to β-smoothness.

4.1 Sublinear convergence

When *no structural assumptions on the functions are made*, sublinear convergence rates typically amount to a $O(1/k)$ global (or $O(1/N)$ ergodic) rate in function values, and a corresponding $O(1/\sqrt{k})$ in the sequence values. More specifically, ADMM converges at a global ergodic rate $O(1/N)$ in function values [67], [43] and at $O(1/\sqrt{k})$ in the sequences of primal and dual variables, for an arbitrarily large stepsize ρ [67].

Since AMA is equivalent to the proximal gradient method applied to the dual function (see Appendix C.1), a $O(1/k)$ convergence rate in the dual function values is proven in [4, Theorem 3.1].

In the case of PDA a partial primal-dual gap function is shown to shrink with rate $O(1/N)$ for the ergodic sequences $\{\bar{z}^N\}$, $\{\bar{\lambda}^N\}$ and $\{\bar{y}^N\}$ in [14].

4.2 Accelerated sublinear convergence

By making further assumptions on the function's structure, faster convergence rates can be recovered. In his work [60], Polyak proposed a way to accelerate the convergence of the gradient method, namely to use the modified update

$$\begin{aligned} \hat{x}^k &= x^k + \alpha^k(x^k - x^{k-1}) \\ x^{k+1} &= \hat{x}^k - \rho^k \nabla f(x^k) \ , \end{aligned}$$

on the smooth convex function f. This method is commonly known as the *heavy ball method*. In this way the next iterate depends on the last gradient update and the previous step $x^k - x^{k-1}$, which is called a *momentum sequence*. This seemingly small change of updating the new iterate as a linear combination of the two previous iterates greatly improves the performance of the original gradient scheme.

In his seminal paper [53], Nesterov modified the heavy ball method by simply evaluating the gradient at the extrapolated point $\hat{x}^k$ instead of x^k. In addition, he proposed a special formula for computing the relaxation sequence $\{\alpha^k\}$, resulting in an *optimal convergence rate for minimizing smooth convex functions using only gradient information.*

The simple update formula is:

$$\alpha^{k+1} = \left(1 + \sqrt{4(\alpha^k)^2 + 1}\right)/2$$

$$\hat{x}^{k+1} = x^k + \frac{\alpha^k - 1}{\alpha^{k+1}}(x^k - x^{k-1}) \tag{4.1}$$

$$x^{k+1} = \hat{x}^{k+1} - \rho^k \nabla f(\hat{x}^{k+1}) \ , \tag{4.2}$$

with $\alpha^0 = 1$. Subsequently, Güler extended Nesterov's results to the proximal point algorithm, handling the minimization of the sum of a smooth and a nonsmooth function [40].

Both ADMM and AMA have benefited from Nesterov's optimal relaxation scheme. Application of the scheme results in an $O(1/k^2)$ global rate of convergence in function values; a rate that is *optimal for first order methods involving a smooth and a nonsmooth function.* Convergence in terms of the sequences is trickier to prove. Same as before and informally speaking, when the optimal $O(1/k^2)$ rate in terms of the primal (dual) function values is achieved, the primal (dual) sequences converge with rate $O(1/k)$ [37],[67], [14]. Apart from the theoretical results, the scheme has been observed to practically accelerate convergence in numerous problem instances. In addition, the extra computational cost is insignificant. PDA also comes with an accelerated variant of global $O(1/k)$ convergence rate of the $\{z^k\}$ sequence, proven in [11, Theorem 19]. The idea for acceleration is quite different from Nesterov's, since it is based on varying stepsizes and a varying relaxation parameter. A local $O(1/k^2)$ convergence in function values also applies, as proven in [14, Theorem 2] in the case that $M = 1$ in Algorithm 5.

ADMM A fast version of ADMM (FADMM), based on Nesterov's acceleration, was first presented in [37]. The algorithm is presented below. Nesterov's optimal relaxation is applied to the sequences $\{y^k\}$ and $\{\lambda^k\}$. The authors use an *adaptive restarting scheme* [56] based

[1] $E^k = \max\{\|s^{k-1}\|, \|r^{k-1}\|\} - \max\{\|s^k\|, \|r^k\|\}$, where s^k and r^k are primal and dual residuals, defined in Appendix A.

Algorithm 6 Fast alternating direction method of multiplier (FADMM)

Require: Initialize $\alpha^0 = 1$, $\hat{y}^0 = y^{-1} \in \mathbb{R}^p$, $\hat{\lambda}^0 = \lambda^{-1} \in \mathbb{R}^p$, and $\rho > 0$

loop

1: $z^k = \underset{z}{\operatorname{argmin}} \quad f(z) + \sum_{i=1}^{M} \left\langle \hat{\lambda}_i^k, T_i z \right\rangle + \frac{\rho}{2} \sum_{i=1}^{M} \|T_i z + t_i - \hat{y}_i^k\|^2$

2: $y_i^{k+1} = \mathbf{prox}_{\frac{1}{\rho} g_i} \left(T_i z^k + t_i + \hat{\lambda}_i^k / \rho \right),\ i = 1, \ldots, M$

3: $\lambda_i^k = \hat{\lambda}_i^k + \rho(T_i z^k + t_i - y_i^k),\ i = 1, \ldots, M$

4: if $E^k > 0$[1]then

5: Apply (4.1) to $y^k, \lambda^k \ \Rightarrow\ \hat{y}^{k+1}, \hat{\lambda}^{k+1}$

6: else

7: $\alpha^{k+1} = 1$, $\hat{y}^{k+1} = y^k$ and $\hat{\lambda}^{k+1} = \lambda^k$

end loop

on the residuals' error in Step 5. Since the momentum sequences often exhibit an oscillatory behavior and might overshoot the optimal values, a check is performed, and if the residuals increase in two subsequent iterations, the acceleration scheme is reset.
In the absence of assumptions, FADMM has a local $O(1/k^2)$ convergence rate. All details are given in [37]. Note that FADMM can be applied to the same family of problems as ADMM with no extra assumptions and small additional computational cost.

FAMA The accelerated version of AMA makes use of Nesterov's acceleration scheme on the dual sequence $\{\lambda^k\}$ [37]. Under the same stepsize restriction as in the basic version, *convergence of the dual objective value* at rate $O(1/k^2)$ has been proven. In the same way that AMA is equivalent to the proximal gradient method applied to the dual problem, FAMA is equivalent to the accelerated version of the proximal gradient method, also known as *Fast Iterative Shrinkage-Thresholding Algorithm (FISTA)* [4], applied to the dual problem. A rate of $O(1/k)$ of the primal sequence is also proven recently in [5], where FAMA is introduced under the name *Fast Dual Proximal Gradient algorithm (FDPG)*. Same as with FADMM, FAMA can practically be applied to every problem that AMA can solve.

Algorithm 7 Fast alternating minimization algorithm (FAMA)

Require: Initialize $\alpha^0 = 1$, $\hat{\lambda}^0 = \lambda^{-1} \in \mathbb{R}^p$, and $\rho \leq \frac{\sigma_f}{\|T\|^2}$

loop

1: $z^{k+1} = \underset{z}{\operatorname{argmin}} \quad f(z) + \sum_{i=1}^{M} \left\langle \hat{\lambda}_i^k, T_i z \right\rangle$

2: $y_i^{k+1} = \mathbf{prox}_{\frac{1}{\rho} g_i} \left(T_i z^{k+1} + t_i + \hat{\lambda}_i^k / \rho \right),\ i = 1, \ldots, M$

3: $\lambda_i^{k+1} = \hat{\lambda}_i^k + \rho(T_i z^{k+1} + t_i - y_i^{k+1}),\ i = 1, \ldots, M$

4: Apply (4.1) to $\lambda^k \ \Rightarrow\ \hat{\lambda}^{k+1}$

end loop

PDA The PDA algorithm comes with an accelerated variant, under the assumption that f is σ_f-strongly convex, denoted hereafter as PDAII. The acceleration is achieved by means of adaptive change of the primal and dual stepsizes τ and ρ_i. Furthermore, instead of taking a fixed momentum sequence $\{2(z^{k+1} - z^k)\}$ as in Algorithm 5, a *variable under-relaxation parameter* θ is introduced. The shceme results in a *global* $O(1/k)$ *convergence rate for the primal sequence* $\{z^k\}$, [11, Theorem 19]. Note that the varying stepsize will alter the quadratic

Algorithm 8 Primal-Dual Algorithm II (PDAII)

Require: Initialize $\lambda^0 \in \mathbb{R}^p$, $z^0 \in \mathbb{R}^n, y^0 \in \mathbb{R}^m$. Choose stepsizes $\tau, \rho_1, \ldots, \rho_{M+1} > 0$ such that $\tau^0 \left(\sum_{i=1}^{M} \rho_i^0 \|T_i\|^2 + \rho_{M+1}^0 \|A\|^2 \right) \leq \sqrt{1 + 2\tau^0 \sigma_f / \mu}$ and $\mu \geq 1$, $\theta^0 = 1/\sqrt{1 + 2\tau^0 \sigma_f / \mu}$.

loop

1: $z^{k+1} = \underset{z \in Z}{\operatorname{argmin}} \quad (1/2) z^T Q z + c^T z + \sum_{i=1}^{M} \left\langle z, T_i^T \lambda_i^k \right\rangle + \left\langle z, y^k \right\rangle + (\mu / 2\tau^k) \|z - z^k\|^2$

2: $\hat{z}^{k+1} = z^{k+1} + \theta^k (z^{k+1} - z^k)$

3: $\lambda_i^{k+1} = \mathbf{prox}_{\rho g_i^\star} \left(\lambda_i^k + \rho_i (T_i \hat{z}^{k+1} + t_i) \right),\ i = 1, \ldots, M$

4: $y^{k+1} = \mathbf{prox}_{\rho S_{\mathcal{D}}} \left(y^k + \rho_{M+1} \hat{z}^{k+1} \right)$

5: $\tau^{k+1} = \theta^k \tau^k$, $\theta^{k+1} = 1/\sqrt{1 + 2\tau^{k+1} \sigma_f / \mu}$,
$\rho_i^{k+1} = \rho_i^k / \theta^{k+1},\ i = 1, \ldots, M+1$

end loop

term in Step 1. In case that Q is diagonal, however, there is no extra computational cost.

4.3 Linear convergence

The linear rate's advantage over the sublinear one is that any accuracy level can be achieved (at least in theory). In practice, the slope of the linear rate can be arbitrarily bad and often sublinear methods behave better than linear ones. *Linear convergence rates can be achieved by making use of the basic versions of the algorithms with the extra assumption of strong convexity of at least one of the functions in the objective*. In other words, linear convergence emerges from *the structural properties* of the functions, if these properties are there. This stands in stark contrast to the accelerated versions of the algorithms presented above that require no extra assumptions, achieving the acceleration only by means of the injected relaxation sequences and variable stepsizes.

Linear convergence of ADMM has been recently proven for several problem formulations. In [21], the authors prove *global linear* $O(1/\omega^k),\ \omega > 0$ *convergence* of both ADMM and PADMM under a variety of scenarios in which at least f is strongly convex and smooth.

AMA also has a linearly convergent version under the extra assumption that *the dual function* $f^\star(T^T\lambda)$ *is strongly convex in the variable* λ (see [69, Proposition 2]). In this scenario, the sequences converge to $z^k \to z^\star$, $y^k \to Tz^\star + t$ and $\lambda^k \to \lambda^\star$ linearly. The assumption can be quite restrictive in control problems since T is (almost) always a tall matrix.

Chambolle and Pock suggest a third algorithm in their paper, *which achieves global linear convergence of both the primal and dual sequences* $\{z^k\}$ *and* $\{\nu^k\}, \{\lambda^k\}$ [14, Theorem 3]. Strong convexity of the functions f and $g^\star$ $(M = 1)$ is required as well as knowledge of the convexity modula. Strong convexity of the conjugate function translates to smoothness of the original ones (g), an assumption that is highly unlike to hold in our problems of interest. The result in generalized in the case of $M > 1$ functions in [11, Theorem 24].

In Table 4.1 we provide an up-to-date report of the existing convergence rates of the methods and their accelerated variants. Wherever a dash '-' appears, it means that there does not exist (or we are not aware of) such a result. In some cases, there might be recent advancements that outperform the results presented here.

	Stepsize restrictions	Strong convexity assumptions	Decouples variables of linear constraints	Convergence in function values	Convergence in sequences
ADMM	no	no	no	ergodic $O(1/k)$ [43], [67]	$O(1/\sqrt{k})$ [67], linear[1] [21], [50], [31]
AMA	yes	yes on $f(z)$	yes	-	$O(1/\sqrt{k})$ on the primal [5], linear[2]
PDA	yes	no	yes	ergodic $O(1/k)$ in partial primal-dual gap [14, Theorem 1], [71]	linear[3] [14, Theorem 3],[11, Theorem 24]
FADMM	no	no	no	$O(1/k^2)$ locally on the dual	-
FAMA	yes	yes on $f(z)$	yes	$O(1/k^2)$ on the dual [37], [5]	$O(1/k)$ on the primal [5]
PDAII	yes	yes on $f(z)$	yes	-	local $O(1/k)$ on the primal [14, Theorem 2], global in [11, Theorem 19]

Table 4.1

[1]Strong convexity of the dual function.
[1]Strong convexity of the dual function.

5

Stepsize selection and Preconditioning

In this chapter we discuss a crucial issue that affects every first order scheme, namely how the limited information provided by a first-order oracle can be optimally used in order to speedup the algorithmic progress. The discussion boils down to two topics: stepsize selection and conditioning of the problem. Although seemingly different, these two aspects are strongly related. We first explore how the stepsize can be selected for the three methods we have presented, moving from well-behaved functions with strong regularity properties to functions for which very little information is provided. We subsequently generalize the notion of the stepsize and show how to select the right metric, *i.e.*, the right space in which the problem should be solved. This is equivalent to preconditioning of the problem.

Although in most of the cases the aforementioned schemes aim at maximizing some proven convergence rate, they are often substituted by heuristics that do not come with the same theoretical performance guarrantees. Nonetheless the benefit might be of more practical importance.

5.1 Stepsize

As in every first order method, the stepsize is crucial for the speed of the convergence. In augmented Lagrangian methods, where the stepsize coincides with the penalty parameter and can be practically unbounded, the question of selecting a suitable one remains open in the general case. We first provide a few results that have to do with optimally choosing the stepsize under some assumptions on the problem structure. We then move to the more general (and practical) case where we do not know much about the problem structure and hence we adapt the stepsize according to the latest information we get from querying the function.

5.1.1 Optimal stepsize selection

The case where the composite term $f(z)$ in the objective is strongly convex and smooth is well-studied and has provided with strong results of linear convergece, as was discussed in Section 4.3. Based on these results, the stepsize parameters can be tuned so that the corresponding convergence rates are maximized. Existence of such results for the Douglas-Rachford method has resulted to *optimal stepsize computations* for which the iterates of the dual function (D) converge linearly to an optimizer. These computations, however, require knowledge of both the strong convexity modulo and the smoothness constant of the function. These quantities are, generally, difficult to recover. Thus, the majority of the authors treat the case of QPs, where both constants are associated to the extreme eigenvalues of the Hessian of (D).

In the case of ADMM, the authors of [31] recover the optimal stepsize ρ, having first proven a linear rate of convergence for inequality form, strongly convex QPs, and under an additional assumption that the constraint matrix is either full row rank or invertible.These requirements are relaxed in the recent works [62] and [33].

Chambolle and Pock's third scheme picks the optimal values for the primal and dual stepsizes τ and ρ as they appear in Algorithm 5, provided that we have two functions in the objective (*i.e.*, $M = 1$) and that both of them are strongly convex. The optimal stepsizes are again

chosen based on the convexity modula. The result is generalized in the case of multiple functions g_i in [11].

5.1.2 Practical stepsize selection

In most cases the problem does not have a favorable structure and/or there is absence of information needed to optimally compute the stepsize. Thereby, we resort to more practical schemes in order to speedup the convergence. Two approaches have been mostly followed in literature to address this issue.

The first approach involves heuristic schemes that give rise to suboptimal fixed stepsizes. The procedures that are followed approximate the ones applied when the problem indeed has the desirable structural properties. Some knowledge of the geometry is still needed, restricting these methods mostly to QPs. These schemes will be discussed in more detail in the next section.

The second approach is adaptive updating of the stepsize(s) based on new information that becomes available as the algorithms iterate. This is commonly achieved by trying to guess the local structural properties of the involved functions either by direct function evaluations, or indirectly, via, *e.g.*, the residuals' evolution. As mentioned in Section 2.3, splitting algorithms converge when consensus to a common solution between the subproblems has been achieved. A good indication for this is the convergence of the primal and dual residuals r^k and s^k (Appendix B). Since the faster the residuals decrease the faster the termination of the algorithm, it intuitively makes sense to try and balance the residuals so that they converge at the same speed. Furthermore, computation of the residuals per iteration exhibit how they evolve, and thus it makes sense to try and 'correct' their behaviour if it is not desired. This can be performed via the stepsizes.

Note that in model predictive control and the associated problems of interest the existence of a quadratic term in the objective is usual. Consequently, although problems of this form might be uncommon in general, we are interested in such structures and will return to them in the preconditioning section.

ADMM Observing the ADMM iterates (Algorithm 3), one easily identifies that the dual stepsize or penalty parameter ρ is nothing but a weight on the penalization of the primal feasibility condition $y-Tz-t$, *i.e.*, a weight on the primal residual r^k (B.4). Consequently, by monitoring the primal residual one can either increase ρ when it gets big or decrease it when it is small in comparison to s^k. This is exactly the scheme proposed in [44], that reads as follows:

$$\rho^{k+1} = \begin{cases} 2\rho & \|r^k\| > c\|s^k\| \\ 0.5\rho & \|s^k\| > \|r^k\|/c, \\ \rho^k & \text{otherwise} \ , \end{cases} \tag{5.1}$$

with $c > 1$. The numbers suggested above are indicative and can be scaled according to the problem. This adaptive stepsize scheme is proven to converge and can frequently reduce the required number of iterations in practical applications.

AMA Although in the original version of AMA, as presented in [69], stepsize selection rules are not explicitely discussed, the equivalence of the method (and its accelerated version FAMA) with the ISTA (and FISTA) algorithms [4], [5] allow for variable stepsizes. This is achieved by means of a *backtracking stepsize rule*. Backtracking is highly used in practical optimization for computing a suitable stepsize without having much knowledge on the structure of the problem [55, Chapter 3]. The approach is the following:

First, a quadratic model that upper bounds the inverse dual function is constructed, *i.e.*,

$$Q_L(\lambda,\mu) = f^\star(\mu)+\langle t,\mu\rangle+\langle\lambda-\mu,\nabla f^\star(\mu)\rangle+\frac{L}{2}\|\lambda-\mu\|^2+G(\lambda) \ , \tag{5.2}$$

where we assumed the existence of only one function G for simplicity. Note that the case where $G(\lambda =)\sum_{i=1}^M G_i(\lambda_i)$ is considered follows in the same way since the λ_i updates happen in parallel. The solution of (5.4) is denoted as

$$p_L(\mu) = \mathbf{prox}_{G/L}\left(\mu - \frac{1}{L}\nabla f^\star(\mu)\right) \ ,$$

which is the proximal iteration of Algorithm 10 in Appendix C.1. A sufficient condition for convergence of the ISTA algorithm is that

$-d(p_L(\mu)) \leq Q_L(p_L(\mu), \mu)$ [4, Lemma 2.3], *i.e.*, the original function evaluated at the next iterate is below the corresponding value of the quadratic model. Subsequently, an *estimate of the local Lipschitz constant* $\bar{L}$ can be computed at every iteration such that the condition is satisfied. The scheme can give rise to significantly smaller estimates of the Lipschitz constant that the conservative upper bould L, and, hence results to larger stepsizes $\rho^k = 1/\bar{L}^k$. The same backtracking rule applies to FISTA, and thus to the FAMA algorithm.

PDHG Same as with ADMM, PDHG algorithm's primal and dual residuals are inversely propotional to the (primal and dual) stepsizes τ and ρ, as indicated in (**??**). Consequently, one can achieve some residual balancing, and thus faster convergence, by adaptively choosing the stepsizes based on the latest residuals' update. The authors of [38] suggest two such schemes. The first scheme assumes knowledge of the quantity $\|(T,\ A)\|$ and the stepsizes are updated in a way such that the stability condition $\tau^k \rho^k \|(T,\ A)\|^2 < 1$ is always satisfied. The second scheme uses a backtracking condition in order to guarantee convergence. In the case of PDHG convergence is based on ensuring that the primal and dual sequences move toward the solution set on every iteration, hence we have monotonicity of the sequences. Roughly speaking, the backtracking scheme reduces the stepsizes when this monotonic decrease is about to be violated. It is quite useful in practice since it often leads to long stepsizes that violate the stability condition. The full details can be found in [38, Algorithms 2 and 3].

Remark 5.1. Another way to understand the role of the primal and dual stepsizes is to observe from Algorithm 5 that they correspond to the weights of the proximal terms. The term $(1/2\tau)\|z - z^k\|^2$, for example, shows that with an increased stepsize τ the new z-iterate is allowed to move further away from the previously computed one. It is evident that penalization weights and stepsizes are different names for describing the same quantity, as happens with ρ in ADMM.

5.2 Metric selection - Preconditioning

The most ponounced weakness of general first-order methods is their inability to efficiently deal with ill-conditioned problems. A change in the metric amounts to substituting the standard Euclidean inner product with a new one, making use of a matrix $V \in \mathbb{S}^n_{++}$ such that

$$\langle x, y \rangle_V = x^T V y, \quad \|x\|_V^2 = x^T V x \ .$$

In this way a change in the coordinate system and metric are performed so that the underlying algorithm adjusts better to the geometry of the problem. This results to faster convergence and robustness against ill-conditioning. The most common example of schemes that perform the optimization in a different metric are *higher order optimization methods* (Newton, Quasi-Newton, SR1 etc.) that practically use exact or approximate Hessians to compute better descent directions [28].

In our framework, we are interested in selecting a space so *that the computational complexity of the problem is not altered*. Given a vector space equipped with the inner product $\langle \cdot, \cdot \rangle_V$, $V \succ 0$, the proximal step of a splitting scheme would change to

$$\mathbf{prox}_{(V)^{-1} f}(x^k) = \inf_{y \in Y} \left\{ f(y) + \frac{1}{2} \|y - x^k\|_V^2 \right\} \ .$$

XXX stepsize notion generalization

Note that a general unstructured V would result to a proximal step that cannot be computed in closed form. Effort has been put into variable metric schemes that enable a simple computation of the proximal step ([10], [6]).

5.2.1 Quadratic functions

XXX

ADMM The most comprehensive work in terms of metric selection for ADMM is [33], thus we briefly present the results here. As we discussed earlier, there exist linear convergence results along with explicitely computed convergence rates. These results hold for the special

case of smooth, strongly convex functions $f(\cdot)$ with full row rank matrix T. The linear convergence rate concerns the dual sequence $\{\lambda^k\}$ and is the result of applying the Douglas-Rachford algorithm XXX on the dual problem (D). More specifically, if f is $\lambda_{\min}(H)$-strongly convex and $\lambda_{\max}(M)$-smooth, the convergence rate is faster for ratios $\frac{\lambda_{\max}(TH^{-1}T^T)}{\lambda_{\min}(TM^{-1}T^T)}$. Note that the above expression corresponds to some notion of condition number, formulated as the ratio of the maximum eigenvalue of a quadratic upper bound to the minimum eigenvalue of a quadratic lower bound for the function f. It is indeed the case that the dual function $d(\lambda)$ is $\lambda_{\min}(TM^{-1}T^T)$-strongly convex and $\lambda_{\max}(TH^{-1}T^T)$-smooth [33, Proposition 8].

A resonable thought is to try and manipulate this condition number to achieve faster convergence. To this end, the dual problem (D) can be formulated in a vector space equipped with the inner pruduct $\langle \cdot, \cdot \rangle_V$. This is equivalent to scaling the dual function's argument, which now writes

$$d(V^T\lambda) = -f^\star(T^TV^T\lambda) - \sum_{i=1}^{M} g_i^\star(-V^T\lambda_i) - \left\langle t, V^T\lambda \right\rangle \quad ,$$

and $d(\lambda)$ becomes $\lambda_{\min}(VTH^{-1}T^TV^T)$-strongly convex and $\|VTM^{-1}T^TV^T\|$-smooth. It is apparent that the *change in the metric is equivalent to left precondt*

$$\text{minimize} \quad \frac{\lambda_{\max}(TH^{-1}T^T)}{\lambda_{\min}(TM^{-1}T^T)} \quad . \tag{5.3}$$

Algorithm 9 Variable metric fast alternating minimization algorithm (VMFAMA)

Require: Initialize $\alpha^0 = 1$, $\hat{\lambda}^0 = \lambda^{-1} \in \mathbb{R}^p$.

loop

1: $z^{k+1} = \underset{z}{\operatorname{argmin}} \quad f(z) + \sum_{i=1}^{M} \left\langle \hat{\lambda}_i^k, -T_i z \right\rangle$

2: $\lambda_i^{k+1} = \mathbf{prox}_{Lg_i^\star}\left(\hat{\lambda}_i^k - L(T_i z^{k+1} + t_i)\right),\ i = 1, \ldots, M$

3: $\alpha^{k+1} = (1 + \sqrt{4(\alpha^k)^2 + 1})/2$

4: $\hat{\lambda}^{k+1} = \lambda^k + \frac{\alpha^k - 1}{\alpha^{k+1}}(\lambda^k - \lambda^{k-1})$

end loop

AMA The function f has to be strongly convex and differentiable. The method is called *Generalized fast dual gradient* in [32] and is equivalent to FAMA when choosing $L = \rho I$ (this is shown in Section C of the Appendix). Note that Step 2 of Algorithm 9 is also identical to Step 3 of PDHG (Algorithm 5). The same convergence rates as in FAMA hold, as proven in [5]. The choice $L = \rho I$ amounts to computing a quadratic over-estimator of f so that

$$f(z_1) \leq f(z_2) + \langle \nabla f(z_2), z_1 - z_2 \rangle + \frac{1}{2\rho}\|z_1 - z_2\|^2 \ . \tag{5.4}$$

This is what we have practically seen in all the algorithms we have encountered, all of them having either a constant or a varying stepsize. By generalizing the notion of the stepsize, we allow for a matrix L^{-1} that allows different curvatures in different directions. The author in [32] shows that (5.4) holds for any $L \in \mathbb{S}_+$ such that $L^{-1} \succ TH^{-1}T^T$, where $f(z) = (1/2)z^T H z + c^T z$, $H \succ 0$. Several options exist, *e.g.*, $L^{-1} = \|TH^{-1}T^T\| I$ [63] that is more probable to preserve the simplicity of the proximal step, to the more compicated $L^{-1} = TH^{-1}T^T + 10^{-4} I$, *i.e.*, scaling with a perturbed version of the Hessian of the dual function. This approach practically results to a second order method for solving the dual problem.

5.2.2 General problems

ADMM

AMA

PDHG

5.3 Preconditioning and Scaling

As with all first order methods, scaling of the problem data is crucial when it comes to practical convergence speed. In what follows, we first formulate equivalent representations for the forms (P) and (S). Then we examine several preconditioning/scaling techniques that exist in the literature for each method.

The equivalent scaled formulation of P can be written as

$$\begin{array}{ll} \text{minimize} & (1/2)\tilde{z}^T\tilde{Q}\tilde{z} + \tilde{c}^T\tilde{z} + \sum_{i=1}^{M} g_i(P_{2i}^{-1}\tilde{y}_i) \\ \text{subject to} & \tilde{T}\tilde{z} - \tilde{y} = -\tilde{t} \\ & \tilde{A}\tilde{z} = \tilde{b} \ , \end{array} \qquad (P_s)$$

where

$$\begin{aligned} \tilde{Q} = P_1^{-T} Q P_1^{-1}, \quad \tilde{c} = P_1^{-T} c, \\ \tilde{T} = P_2 T P_1^{-1}, \quad \tilde{t} = P_2 t, \\ \tilde{A} = P_3 A P_1^{-1}, \quad \tilde{b} = P_3 b, \end{aligned}$$

and $P_1 \in \mathbb{R}^{n\times n}$, $P_2 = (P_{21}, \ldots, P_{2M}) \in \mathbb{R}^{p\times p}$, $P_3 \in \mathbb{R}^{m\times m}$.

Equivalently, the saddle function (S) can be expressed as

$$\mathcal{S}_s = \left\langle \tilde{T}\tilde{z} + \tilde{t}, \tilde{\lambda} \right\rangle + \left\langle \tilde{A}\tilde{z} - \tilde{b}, \tilde{\nu} \right\rangle + (1/2)\tilde{z}^T\tilde{Q}\tilde{z} + \tilde{c}^T\tilde{z} - \sum_{i=1}^{M} g_i(P_{2i}^T\tilde{\lambda}_i) \ . \quad (S_s)$$

The original variables (z, y, λ) are associated with the scaled ones $(\tilde{z}, \tilde{y}, \tilde{\lambda})$ via the transformations

$$z = P_1^{-1}\tilde{z}, \quad y = P_2^{-1}\tilde{y}, \quad \lambda = P_2^T\tilde{\lambda}, \quad \nu = P_3^{-1}\tilde{\nu}.$$

There is enough freedom to choose the matrices P_1, P_2, P_3 as we like. We refer to *scaling* of the problem when the matrices are chosen to be diagonal with positive elements. The term *preconditioning* encompasses any general matrix. The purpose of performing this procedure

is to alter the problem data in a way that is favorable for the convergence of the algorithms. The idea originally comes from the need to solve large systems of linear equations by means of iterative methods. The problems we are interested in are not that big, however spitting algorithms can still benefit from data scaling, either by rendering the (scaled) matrices more well-conditioned, and hence more stable to invert, or by altering the spectrum of specific matrices that affect the stepsize selection.

On the other hand, one should ensure that the scaled varsion of the problem does not alter significantly the computational complexity of the subproblems to solve. It is easy to see that for several choices of the matrices P_{2i} in (P_s) the proximal step might not be computable in closed form. Thus any kind of preconditioning should be used with caution.

We present a few existing preconditioning schemes for the algorithms.

ADMM ADMM has no stepsize restrictions. However, as every first order method, is is highly dependent on data values. It is thus adviced to improve the conditioning of the Hessian matrix of the augmented Lagrangian, which can lead to faster convergence. A common (and simple) way to perform matrix scaling is to equilibrate its rows and columns so that their infinity norm gets closer to one [66]. The authors of [16] perform a heuristic method in this direction in their ADMM-like primal-dual scheme. The P_1 and P_2 matrices are chosen as follows.

We define

$$\hat{T}_{ij}^2 = (1/|p_i|) \sum_{k \in p_i} T_{kj}^2, \quad \hat{A}_j^2 = \sum_{k=1}^{m} A_{kj}^2 \ ,$$

and the vectors $p_2 \in \mathbb{R}^M$, $p_3 \in \mathbb{R}$, $p_1 \in \mathbb{R}^n$:

$$p_{2i} = \left(\sum_{j=1}^{n} \hat{T}_{ij}^2 \right)^{-1/2}, \quad p_3 = \left(\sum_{j=1}^{n} \hat{A}_j^2 \right)^{-1/2}, \quad p_{1j} = \left(\sum_{i=1}^{M} \hat{T}_{ij}^2 p_{2i}^2 + \hat{A}_j^2 p_3^2 \right)^{-1/2} .$$

Then the scaling matrices can be constructed as

$$P_2 = \mathbf{diag}(p_{2i} I_{p_i}), \quad P_3 = \mathbf{diag}(p_3 I_m), \quad P_1 = \mathbf{diag}(p_{1j}) \ .$$

In the case that we have more knowledge about the optimization problem at hand, more sophisticated preconditioners can be used. The authors of [31] solve an SDP in order to acquire the optimal preconditioner for strongly convex QPs with full row rank assumption on (T, A).

AMA XXX Ye

PDHG XXX CP

6

Numerical Linear Algebra

7

Summary

8

Examples

We demonstrate some of the methods presented in the previous sections with two optimal control problems. The first example involves MPC for tracking of a reference signal, boiling down to solving a sequence of QPs, while the second example considers the planetary soft landing problem, an originally nonconvex problem that is relaxed to an SOCP. We focus on explaining how to rewrite our problems so that we maximally exploit the ideas presented in Section **??**.

1. **Aircraft control**
   In this example the linearized model of a Boeing 747-200 (B747) is considered [41]. The model has $n = 12$ states and $m = 17$ inputs and the aim is tracking of a reference signal $r(k)$ for three of the states. We discretize with sampling period $T_s = 0.2s$ and consider in total a signal of 115 setpoints. Firstly, a *steady state target calculator* computes a pair of setpoints $(\delta x_s(k), \delta u_s(k))$ for the aircraft, according to a desired reference signal. Subsequently, an MPC controller is tracking the delivered setpoint. The steady-states are generated by solving a strongly convex dense QP with $n + m = 29$ variables and bound constraints on the inputs [41, Section II,B]. The affine term in the objective is a function of $r(k)$, hence the optimization has to be performed as many

times as is the length of the reference signal. The MPC problem is a simple quadratic one, with $Q \succeq 0$ and the same bound constraints on the inputs. The affine term is also time-varying since it is a function of the generated setpoints.

Steady state calculator The problem to solve is

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\theta_s^T H_s \theta_s - h_s(k)^T \theta_s \\ \text{subject to} & \theta_{min} \leq \theta_s \leq \theta_{max} \ , \end{array} \tag{8.1}$$

with variables $\theta_s \in \mathbb{R}^{n+m}$ and $H_s \succ 0$. Since the objective is strongly convex, we can use accelerated versions of the methods. To this end, FAMA and CPII are valid options, however, the dense structure of H_s would require a forward backward substitution at each iteration, something that can be avoided. We thus take the Cholesky factorization of H_s, *i.e.*, $H_s = LL^T$, L is lower triangular and invertible and perform a change of basis, $\tilde{\theta}_s = L^T \theta_s$. Now the problem can be reformulated as

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\tilde{\theta}_s^T \tilde{\theta}_s - \tilde{h}_s(k)^T \tilde{\theta}_s \\ \text{subject to} & C\tilde{\theta}_s \leq d \ , \end{array} \tag{8.2}$$

with variables $\tilde{\theta}_s \in \mathbb{R}^{n+m}$, $\tilde{h}_s(k) = L^{-1} h_s(k)$. The matrix-vector pair (C, d) describes the polytopic constraints that are now imposed in the place of the simple bound constraints that we had in (8.1). This is the price paid for eliminating the dense Hessian in the objective. By introducing a slack variable $y = C\tilde{\theta}_s - d$, $y \leq 0$, we can apply FAMA to the modified problem with $f(\tilde{\theta}_s) = \frac{1}{2}\tilde{\theta}_s^T \tilde{\theta}_s - \tilde{h}_s(k)^T \tilde{\theta}_s$, $l(y) = \delta_-(y)$, $T = C$, $t = -d$. For the stepsize we choose $\rho = 1/\lambda_{max}(C^T C)$.

As a second option, we use ADMM with the parameters tuned as in [31] in the same setting. This version achieves linear convergence rate by means of the optimal stepsize selection $\rho = 1/\sqrt{\lambda_{min}(CC^T)\lambda_{max}(CC^T)}$. In our case C is singular and so we consider the smallest nonzero eigenvalue.

Accordingly we can use CPII. Problem 8.2 can be written in a saddle

point form as

$$\min_{\tilde{\theta}_s} \max_{\lambda} \left\{ \left\langle C\tilde{\theta}_s - d, \lambda \right\rangle + \frac{1}{2}\tilde{\theta}_s^T \tilde{\theta}_s - \tilde{h}_s(k)^T \tilde{\theta}_s - \delta_+(\lambda) \right\} ,$$

so we can use CPII with $Z = \mathbb{R}^{n+m}$, $g_i^\star(\lambda) = \delta_+(\lambda)$, T, t as defined above. Note that there are no equality constraints, hence there is no ν-update. We initialize the primal stepsize $\tau^0 = 100$ according to [14, Theorem 2].

We solve the problem 115 times with the affine term varying slightly from one iteration to the other. We terminate based on the residual decrease, with the accuracy threshold set to 10^{-3} for FAMA and CPII and 10^{-4} for ADMM (see Remark **??**). FAMA needs 495 iterations on average, with average time 0.85ms per solve, ADMM 194 iterations at 0.56ms per solve and CPII 1100 iterations at 4.9ms per solve. The solutions achieved are quite accurate, with a normed relative error ($\|\theta_s - \theta_s^\star\|/\|\theta_s^\star\|$) of $\approx 10^{-5}$ for all the methods, sumed over all 115 instances. The optimal stepsize selection renders ADMM clearly superior in this case.

MPC for tracking The MPC problem described in [41, Section II] can be written in the condensed form

$$\begin{array}{ll} \text{minimize} & \delta_{u_s}^T \delta_{u_s} + h(k)^T \delta_{u_s} \\ \text{subject to} & C\delta_{u_s} \leq d \ , \end{array} \tag{8.3}$$

with variables $\delta_{u_s} \in \mathbb{R}^{Nm}$, after having changed the basis in the same way as before. We solve the problem for the following scenarios: $N = 5$, cold start, warm started at the primal and dual optima of the previous solve. The outputs are reported in Table 8.1. We use the same normed relative error $\|(x,u) - (x^\star, u^\star)\|/\|((x^\star, u^\star))\|$ to evaluate the quality of the solution. ADMM behaves significantly better than the other two methods in terms of iterations, but FAMA is faster overall in timings. With the number of variables increasing, the cost per iteration starts being more evident when using ADMM. We observe that warm starting makes a big difference in terms of iteration counts.

2. **The planetary soft landing problem**
The problem presented in [2] regards the situation where an au-

		ADMM	FAMA	CPII
$N = 5$	Av. No. Iters. Cold\Warm	1362 \548	2279 \778	1544\825
	Min.\Max. No. Iters. Warm	72\1504	83\5947	1\2111
	Av. Time Cold \Warm (ms)	46.90\19.82	42.74 \14.82	75.16 \40.53
	Relative error	1.61×10^{-4}	1.62×10^{-4}	1.61×10^{-4}
$N = 12$	Av. No. Iters. Cold\Warm	734\422	2999\1150	1721\1053
	Min.\Max. No. Iters. Warm	171\1079	167\10093	1\3389
	Av. Time Cold \Warm		274.54 \110.27	430.25 \266.75
	Relative error	4.7×10^{-5}	4.6×10^{-5}	4.6×10^{-5}

Table 8.1

tonomous spacecraft lands on the surface of a planet by using thrusters, which produce a force vector that has both an upper and a nonzero lower bound on its magnitude. The control constraints are thus represented by a nonconvex set which has the form of a ring. This kind of constraints appears in a plethora of optimal control problems, but the case of planetary landing is of interest because, under some assumptions, the optimal trajectories of a relaxed version of the problem (where the nonconvex constraint set is replaced with a convex one) are also optimal for the original problem. Hence a lossless convexification can be achieved. The relaxed problem results in being an SOCP.

We first present the original problem formulation:

$$\begin{array}{ll} \text{minimize} & (x_0 - z_0)^T Q (x_0 - z_0) + \alpha \sum_{i=0}^{N-1} \|u_i\|_2 \\ \text{subject to} & x_{i+1} = Ax_i + Bu_i + Ew_i,\ i = 0, \ldots, N-1 \\ & x_N = z_f, \\ & \gamma |e_1^T x_i| \leq e_2^T x_i,\ i = 0, \ldots, N \\ & 1 \leq \|u_i\| \leq c,\ i = 0, \ldots, N-1\ , \end{array} \tag{8.4}$$

with variables x_i, u_i and $\alpha \in \mathbb{R}_{++}$ a positive weight. Variable $x = (p_x, p_y, v_x, v_y)$ is the state of the $x - y$ position and the corresponding velocity coordinates of the spacecraft. The conic constraint correspondes to a *glide slope* constraint, ensuring that the spacecraft remains in a cone defined by a minimum slope angle. The nonconvex constraint on the inputs can be convexified by lifting, as discussed in [2, Section 3]. The relaxed problem can be written as

$$\begin{array}{ll} \text{minimize} & (x_0 - z_0)^T Q (x_0 - z_0) + \alpha \sum_{i=0}^{N-1} \sigma_i \\ \text{subject to} & x_{i+1} = Ax_i + Bu_i + Ew_i,\ i = 0, \ldots, N-1 \\ & x_N = z_f, \\ & \gamma |e_1^T x_i| \leq e_2^T x_i,\ i = 0, \ldots, N \\ & \|u_i\| \leq c,\ i = 0, \ldots, N-1 \\ & \|u_i\| \leq \sigma_i,\ i = 0, \ldots, N-1 \\ & \sigma_i \geq 1,\ i = 0, \ldots, N-1\ , \end{array} \tag{8.5}$$

with variables x_i, u_i, σ_i.

There now exist several scenaria in which the solutions of (8.5) and (8.4) coincide. We consider one such scenario, similar to the numerical instances presented in [2, Section 5]. The data are

$$A = \begin{bmatrix} 0 & I \\ -\theta^2 I & \theta S \end{bmatrix} \in \mathbb{R}^n,\ B = E = \begin{bmatrix} 0 \\ I \end{bmatrix} \in \mathbb{R}^m\ ,\ S = \begin{bmatrix} 0 & 2 \\ -2 & 0 \end{bmatrix}\ ,$$

$g = 3.7114$ is the gravitational acceleration of Mars, $w = -e_2 g$, $\theta = 1/(1.02595675 \times 24 \times 60 \times 60)$ is its rotation rate, $c = 4$, $\gamma = 1/\sqrt{3}$, $z_0 = (-15, 20, -10, 1)$, $z_f = (0, 0.1, 0, 0)$. The system is discretized using a zero-order hold with sampling period of $0.33s$. For a final time of $15s$ this gives $N = 45$. The weight matrix that penalized the initial

state is chosen as $Q = \mathbf{diag}(2, 2, 1, 1)$.

Problem's (8.5) objective value does not exhibit any favorable characteristics (strong convexity or smoothness), and thus cannot benefit from an accelerated version of the methods mentioned above. In this case, we prefer to increase the number of slack variables and benefit from the simplicity of the prox operators. We first consider the augmented Lagrangian formulation AL with the following functions:

$$f(x, u, \sigma) = \left\{ (x_0 - z_0)^T Q (x_0 - z_0) + \sum_{i=0}^{N-1} \sigma_i \colon \boldsymbol{A}(x, u) = \boldsymbol{b} \right\} \tag{8.6}$$

$$g_i^1(Gx_i) = \delta_2(e_1^T x_i, (1/\gamma) e_2^T x_i),\ i = 0, \ldots, N \tag{8.7}$$

$$g_i^2(u_i) = \delta_2(u_i, c),\ i = 0, \ldots, N-1 \tag{8.8}$$

$$g_i^3(u_i, \sigma_i) = \delta_2(u_i, \sigma_i),\ i = 0, \ldots, N-1 \tag{8.9}$$

$$g_i^4(\sigma_i) = \delta_+(\sigma_i - 1),\ i = 0, \ldots, N-1\ . \tag{8.10}$$

We overloaded notation for the dynamics' equation by denoting

$$\boldsymbol{A} = \begin{bmatrix} -A & -B & I & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & -A & -B & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \\ 0 & 0 & 0 & 0 & \cdots & I & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & -A & -B & I \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & I \end{bmatrix}, \qquad \boldsymbol{b} = \begin{bmatrix} Ew_0 \\ Ew_1 \\ \vdots \\ Ew_{N-1} \\ z_f \end{bmatrix},$$

and G is defined as $G = \begin{bmatrix} e_1^T \\ (1/\gamma) e_2^T \end{bmatrix}$. One can see that due to the several constraints, many copies of the variables have to be introduced. We denote the slack variables and the corresponding multipliers as

$$\begin{array}{cc} y_i = (y_i^1, y_i^2) = Gx_i, & \text{with multiplier } \lambda_i^y = (\lambda_i^{y^1}, \lambda_i^{y^2}) \in \mathbb{R}^2 \text{ for (8.7).} \\ u_i = \tilde{u}_i, & \text{with multiplier } \tilde{\lambda}_i^u \text{ for (8.8).} \\ (u_i, \sigma_i) = (\hat{u}_i, \hat{\sigma}_i), & \text{with multiplier } \hat{\lambda}_i = (\hat{\lambda}_i^u, \hat{\lambda}_i^\sigma) \in \mathbb{R}^{m+1} \text{ for (8.9).} \\ \sigma_i = \tilde{\sigma}_i, & \text{with multiplier } \tilde{\lambda}_i^\sigma \text{ for (8.10). .} \end{array}$$

The steps of ADMM as presented in Algorithm 3 are summarized below.

- The first step in the algorithm involves the minimization of (8.6). This optimization problem can be further split in two steps, a linearly constrained QP with variables (x, u) and an unconstrained QP for σ_i, decomposed for $i = 0, \ldots, N-1$. The first QP can be expressed as

$$\begin{array}{ll} \text{minimize} & (1/2)s^T H s + h^T s \\ \text{subject to} & \boldsymbol{A}s = \boldsymbol{b}, \end{array} \tag{8.11}$$

over variable $s \in \mathbb{R}^{(N+1)n+Nm}$, where in the objective we group only quadratic and affine terms and the equality constraint corresponds to the dynamics of the system (see also [57]). We denote the following:

$$s = \begin{bmatrix} x_0 \\ u_0 \\ \vdots \\ u_{N-1} \\ x_N \end{bmatrix}, \qquad h = \begin{bmatrix} -G^T(\rho y_0^k + (\lambda_0^y)^k) - 2Qz_0 \\ \rho(\tilde{u}_0^k + \hat{u}_0^k) + (\tilde{\lambda}_0^k + \hat{\lambda}_0^k) \\ \vdots \\ -G^T(\rho y_N^k + (\lambda_N^y)^k) \end{bmatrix},$$

$$H = \begin{bmatrix} 2Q + \rho G^T G & 0 & 0 & \cdots & 0 & 0 \\ 0 & 2\rho I & 0 & \cdots & 0 & 0 \\ 0 & 0 & \rho G^T G & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 2\rho I & 0 \\ 0 & 0 & 0 & \cdots & 0 & \rho G^T G \end{bmatrix},$$

and

The update is the solution of the linear system

$$\begin{bmatrix} H & \boldsymbol{A}^T \\ \boldsymbol{A} & 0 \end{bmatrix} \begin{bmatrix} s \\ \nu \end{bmatrix} = \begin{bmatrix} -h \\ \boldsymbol{b} \end{bmatrix}, \tag{8.12}$$

The σ-update has the closed form solution

$$\begin{aligned} \sigma_i^{k+1} =& \frac{1}{2}\left(\hat{\sigma}_i^k + \tilde{\sigma}_i^k + (1/\rho)((\hat{\lambda}_i^\sigma)^k + (\tilde{\lambda}_i^\sigma)^k - \alpha)\right), \\ & i = 0, \ldots, N-1 \ . \end{aligned} \tag{8.13}$$

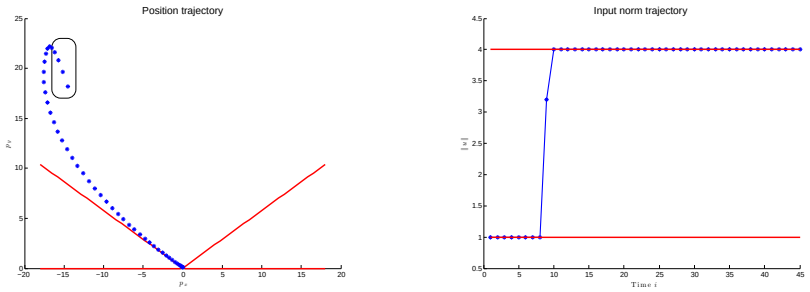
	ADMM	FADMM	aPDHG
No. Iters.	1224	1501	7598
Time (ms)			
Relative error	9.80×10^{-3}	10.70×10^{-3}	35.90×10^{-3}

Table 8.2

- The second step invlolves a series of proximal minimization problems for each of the introduced variables (equations (8.7),(8.8),(8.9),(8.10). All the updates can be expressed in closed form and are separable across time. They involve projections on the nonnegative orthant, the second-order cone and the 2-norm ball. We will not write down the analytic expression for each one of the updates, but they can be easily calculated by consulting Table XXX

  Another option is to go for the PDHG algorithm, which again has no assumptions on the problem structure. In this case we use the adaptive stepsize PDHG method (Algorithm **??**), which is known to work better in practice than its fixed stepsize counterpart. The steps are very similar to ADMM, with the only differences that 1) The dynamics enter the dual update ν (Step 4 of Algorithm **??**) and 2) All the proximal steps are performed with respect to the conjugate functions, as they are given in Table A.

  Using over-relaxed ADMM and FADMM, the stepsize is chosen to be $\rho = 0.5$ and the over-relaxation parameter at 1.8. For the Adaptive PDHG we set $\tau = 100$. The error tolerances are set to 10^{-3}. The results are reported in Table 8.2. It is surprising that the over-relaxed ADMM bahaves better than the accelerated version in this case. We do not have a decisive argument on why this holds, since FADMM uses a heuristic rule for restarting which does not necessarily make it better in all cases. It seems that the small stepsize we use results to relatively smooth behaviour in



(a) $x - y$ coordinates of spacecraft **(b)** Norm of inputs

Figure 8.1: Trajectory of the position as the spacecraft lands to the specified point $z_f = (0, 0.1)$. The initial state lies within some interval away from the desired one $z_0 = (-15, 20)$. With red color is depicted the glide angle constraint. Observe that many state constraints are active at the optimal trajectory. In the second plot the optimal input trajectory is depicted. As expected, the convex relaxation is exact, *i.e.*, the inputs stay in between 1 and 4. Note that almost all of them are saturated at optimality.

the residuals, and hence the restarting scheme is not frequently activated. Furthermore, once the over-relaxation parameter is set to 1, the number of iterations of ADMM increases. aPDHG converges very slowly, rendering it an unsuitable option in this case. We should mention that this is a specifically difficult problem to solve due to the big number of active constraints at optimality. This is the reason that even augmented Lagrangian-based methods, which usually behave well, need so many iterations for convergence.

Appendices

A

Conjugacy and the Proximal operator

Description	$f(x)$	$f^\star(p)$
Nonnegative orthant	$\delta_+(x)$	$\delta_-(p)$
Semidefinite cone	$\delta_{S_+}(X)$	$\delta_{S_-}(P)$
Box ($\|x\|_\infty \leq l$)	$\delta_\infty(x,l)$	$l\|p\|_1$
l_1-norm ball ($\sum_i \|x_i\| \leq l$)	$\delta_1(x,l)$	$l\|p\|_\infty$
l_2-norm ball ($\|x\|_2 \leq l$)	$\delta_2(x,l)$	$l\|p\|_2$
Lorentz cone ($\|x\|_2 \leq t$, $(x,t) \in \mathbb{R}^{n+1}$)	$\delta_2(x,t)$	$\delta_2(p,y)$

The conjugates are useful for computing the solution of the optimization problem at Step 2 of Algorithm **??**. Step 2 can be alternatively written as

$$p_i^{k+1} = \mathbf{prox}_{\sigma g_i^\star}(\tilde{p}_i^k)\ i = 1, \ldots, M, \tag{A.1}$$

where $\tilde{p}_i^k = p_i^k + \sigma(T_i \hat{z}^k - t_i)$. A very useful result is the Moreau identity [51], stated below.

Lemma A.1. Let $f : \mathbb{R}^n \to (-\infty, \infty]$ be a proper, lsc convex function. Then for any $x \in \mathbb{R}^n$

$$\mathbf{prox}_{\rho f^\star}(x) + \rho\, \mathbf{prox}_{f/\rho}(x/\rho) = x, \quad \forall\ 0 < \rho < +\infty\ .$$

From this Lemma, we can conclude that

$$\mathbf{prox}_{\rho h^\star}(x) = x - \rho\,\mathbf{prox}_{h/\rho}(x/\rho). \tag{A.2}$$

In this way, the solution to the step can directly be computed using the table above.

B

Stopping Conditions

We make use of the KKT conditions in order to terminate the algorithms presented in this work. The procedure is the following: We write down the optimality conditions for problem (P) or for the saddle problem (PD) and express them in terms of the optimality conditions derived from each iterate of the corresponding algorithms. The algorithm is terminated once the KKT conditions are satisfied to some prespecified accuracy. More specifically we have:

ADMM Consider the Lagrangian (L). The KKT conditions are:

$$0 = T_i z^\star + t_i - y_i^\star, \quad i = 1, \ldots, M \tag{B.1}$$

$$0 = \nabla f(z^\star) + \sum_{i=1}^{M} T_i^T \lambda_i^\star \tag{B.2}$$

$$0 \in \partial g_i(y_i^\star) - \lambda_i^\star, \quad i = 1, \ldots, M \tag{B.3}$$

Writing the optimality conditions for each step of Algorithm 3, we derive the formulas for the primal and dual residuals,

$$r^{k+1} = T z^{k+1} + t - y^{k+1}, \tag{B.4}$$

$$s^{k+1} = \rho T^T (y^k - y^{k+1}) \ , \tag{B.5}$$

as given in [13]. If the problem is scaled as presented in Section 5.3, the residuals scale accordingly and become

$$r^{k+1} = P_2^{-1}(\tilde{T}\tilde{z}^{k+1} + \tilde{t} - \tilde{y}^{k+1}) \tag{B.6}$$
$$s^{k+1} = \rho(P_2^{-1}\tilde{T}P_1)^T P_2^{-1}(\tilde{y}^k - \tilde{y}^{k+1}) \ . \tag{B.7}$$

AMA We consider again the KKT conditions (B.1), (B.2) and (B.3). Taking the optimality condition for Step 1 of Algorithm 4, we have that

$$\begin{aligned}
&\nabla f(z^{k+1}) + \sum_{i=1}^{M} T_i^T \lambda_i^k = 0 \\
&\nabla f(z^{k+1}) + \sum_{i=1}^{M} T_i^T \lambda_i^{k+1} + \sum_{i=1}^{M} T_i^T(\lambda_i^k - \lambda_i^{k+1}) = 0 \ ,
\end{aligned}$$

hence condition (B.2) is satisfied if $T^T(\lambda^{k+1} - \lambda^k) = 0$. Accordingly we have for Step 2 that

$$\begin{aligned}
&\partial g_i(y_i^{k+1}) + \lambda_i^k + \rho(T_i z^{k+1} + t_i - y_i^{k+1}) \ni 0 \\
&\partial g_i(y_i^{k+1}) + \lambda_i^{k+1} \ni 0 \ ,
\end{aligned}$$

which means that condition (B.3) is always satisfied each time Step 2 is executed. Finally, the primal optimality conditions reads $T_i z^{k+1} + t_i - y_i^{k+1} = 0,\ i = 1, \ldots, M$. We can thus write the primal and dual residuals as

$$r^{k+1} = Tz^{k+1} + t - y^{k+1} \tag{B.8}$$
$$s^{k+1} = T^T(\lambda^{k+1} - \lambda^k) \ . \tag{B.9}$$

As with ADMM, the scaled residuals become

$$r^{k+1} = P_2^{-1}(\tilde{T}\tilde{z}^{k+1} + \tilde{t} - \tilde{y}^{k+1}) \tag{B.10}$$
$$s^{k+1} = P_1\tilde{T}^T(\tilde{\lambda}^{k+1} - \tilde{\lambda}^k) \ . \tag{B.11}$$

PDA Consider the saddle function (S). The KKT conditions are:

$$0 = Qz^\star + c + \sum_{i=1}^{M} T_i^T p_i^\star + y^\star \tag{B.12}$$

$$0 \in \partial g_i^\star(p_i^\star) - T_i z^\star - t_i, \quad i = 1, \ldots, M \tag{B.13}$$

$$0 = \partial S_{\mathcal{D}}(y^\star) - z^\star \tag{B.14}$$

As before, we write down the optimality conditions for each step of Algorithm 5. The derivation has been performed in [38, Section 2.1] and the residuals read

$$r^{k+1} = \frac{1}{\tau^k}(z^k - z^{k+1}) + T^T(\lambda^{k+1} - \lambda^k) + (y^{k+1} - y^k) \tag{B.15}$$

$$s_1^{k+1} = P^k(\lambda^k - \lambda^{k+1}) + T(z^{k+1} - z^k) \tag{B.16}$$

$$s_2^{k+1} = \frac{1}{\rho_{M+1}^k}(y^k - y^{k+1}) + (z^{k+1} - z^k) \ , \tag{B.17}$$

where $P^k = \mathbf{diag}(\frac{1}{\rho_1^k}, \ldots, \frac{1}{\rho_M^k})$. In the case of the scaled problem the residuals are XXX

$$r^{k+1} = \frac{1}{\tau^k} P_1^{-1}(\tilde{z}^k - \tilde{z}^{k+1}) + P_1^T \begin{bmatrix} \tilde{T} \\ \tilde{A} \end{bmatrix}^T \begin{bmatrix} \tilde{\lambda}^k - \tilde{\lambda}^{k+1} \\ \tilde{\nu}^k - \tilde{\nu}^{k+1} \end{bmatrix} \tag{B.18}$$

$$s^{k+1} = \frac{1}{\rho^k} \begin{bmatrix} P_2 \\ P_3 \end{bmatrix}^T \begin{bmatrix} \tilde{\lambda}^k - \tilde{\lambda}^{k+1} \\ \tilde{\nu}^k - \tilde{\nu}^{k+1} \end{bmatrix} + \begin{bmatrix} P_2^{-1}\tilde{T} \\ P_3^{-1}\tilde{A} \end{bmatrix} (\tilde{z}^k - \tilde{z}^{k+1}) \tag{B.19}$$

PDAII In this case, though the KKT conditions are again (B.12), (B.13), (B.14), the derivation of the residuals is slightly different from the ones in the PDA method due to the time-varying relaxation parameter θ^k. From Step 3 of Algorithm 8 we get:

$$0 \in \partial g_i^\star(p_i^{k+1}) - T_i \hat{z}^{k+1} - t_i + \frac{1}{\rho^k}(\lambda_i^{k+1} - \lambda_i^k).$$

By substituting $\hat{z}^{k+1}$ from Step 2, we have that condition (B.13) holds if

$$\frac{1}{\rho_i^k}(\lambda_i^k - \lambda_i^{k+1}) + \theta^k T_i(z^{k+1} - z^k) = 0$$

Similarly, from Step 4 we have that

$$0 \in \partial S_{\mathcal{D}}(y^{k+1}) - \hat{z}^{k+1} + \frac{1}{\rho_{M+1}^k}(y^{k+1} - y^k),$$

which gives rise to the condition

$$\frac{1}{\rho_{M+1}^k}(y^k - y^{k+1}) + \theta^k(z^{k+1} - z^k) = 0$$

Finally, the optimizer for Step 1 is the same as in the case of PDA. Consequently, we can define the residuals

$$r^{k+1} = \frac{1}{\tau^k}(z^k - z^{k+1}) + T^T(\lambda^{k+1} - \lambda^k) + (y^{k+1} - y^k) \tag{B.20}$$

$$s_1^{k+1} = P^k(\lambda^k - \lambda^{k+1}) + \theta^k T(z^{k+1} - z^k) \tag{B.21}$$

$$s_2^{k+1} = \frac{1}{\rho_{M+1}^k}(y^k - y^{k+1}) + \theta^k(z^{k+1} - z^k) \ , \tag{B.22}$$

In the scaled form we have that XXX

$$r^{k+1} = \frac{1}{\tau^k}P_1^{-1}(\tilde{z}^k - \tilde{z}^{k+1}) + P_1^T \begin{bmatrix} \tilde{T} \\ \tilde{A} \end{bmatrix}^T \begin{bmatrix} \tilde{\lambda}^k - \tilde{\lambda}^{k+1} \\ \tilde{\nu}^k - \tilde{\nu}^{k+1} \end{bmatrix} \tag{B.23}$$

$$s^{k+1} = \frac{1}{\rho^k} \begin{bmatrix} P_2 \\ P_3 \end{bmatrix}^T \begin{bmatrix} \tilde{\lambda}^k - \tilde{\lambda}^{k+1} \\ \tilde{\nu}^k - \tilde{\nu}^{k+1} \end{bmatrix} + \begin{bmatrix} P_2^{-1}\tilde{T} \\ P_3^{-1}\tilde{A} \end{bmatrix} (\tilde{z}^{k+1} - \tilde{\tilde{z}}^{k+1}) \tag{B.24}$$

C

AMA and Dual Proximal Gradient Method

The subsequent result is instrumental as it demonstrates the equivalence of the AMA algorithm with the Proximal Gradient Method applied to the dual problem. It also draws strong connections of AMA with the PDHG algorithm. The approach is inspired from [5, Lemma 3.2], where the equivalence between FAMA and the Dual Fast Proximal Gradient Method is drawn.

Consider again the dual function (D)

$$d(\lambda) = f^\star(-T^T\lambda) + \sum_{i=1}^{M} g_i^\star(\lambda_i) - \langle t, \lambda \rangle \quad .$$

Under the assumption that d is continuously differentiable and ∇d is Lipschitz continuous with constant L, the proximal gradient algorithm writes

Algorithm 10 Dual Proximal Gradient Method

Require: Initialize $\rho = 1/L$.

loop

1: $z^{k+1} = \underset{z}{\operatorname{argmin}} \quad f(z) + \left\langle T^T\lambda^k, z \right\rangle$

2: $\lambda_i^{k+1} = \mathbf{prox}_{\rho g_i^\star}\left(\lambda_i^k + \rho(T_i z^{k+1} + t_i)\right),\ i = 1, \ldots, M$

end loop

The following Lemma then holds:

Lemma C.1. The second step of the Algorithm 10 is equivalent to steps 2 and 3 of Algorithm 4 combined, written as

$$\begin{aligned} y_i^{k+1} &= \mathbf{prox}_{\,g_i/\rho}\left(T_i z^{k+1} + t_i + \lambda_i^k/\rho\right) \\ \lambda_i^{k+1} &= \lambda_i^k + \rho(T_i z^{k+1} + t_i - y_i^{k+1}). \end{aligned}$$

Step 1 of Algorithm 10 amounts to computing the gradient of $f^\star(T^T\lambda) - \langle t, \lambda\rangle$. Since f is strongly convex (the assumption holds for both algorithms), its conjugate is continuously differentiable (see [Corollary 18.16][3]). Consequently,

$$-T^T\lambda^k \in \partial f(z^{k+1}) \Leftrightarrow z^{k+1} = \nabla f^\star(-T^T\lambda^k)\ .$$

Thus, the gradient of the smooth part of the dual objective, denoted as $F(\lambda) = f^\star(T^T\lambda) - \langle t, \lambda\rangle$ writes as

$$\nabla F(\lambda^k) = -Tz^{k+1} - t\ .$$

Step 2 is, apparently, the proximal step in the direction of the negative gradient.

We will finally show that the two steps of AMA are equivalent to Step 2 of Algorithm 10. To this end, we first denote $d_i^k = \lambda_i^k + \rho(T_i z^{k+1} + t_i)$, and subsequently substitute the result of the proximal step y_i^{k+1} to the dual update λ_i^{k+1} :

$$\begin{aligned} \lambda_i^{k+1} &= \lambda_i^k + \rho(T_i z^{k+1} + t_i) - \rho\,\mathbf{prox}_{\,g_i/\rho}(\lambda_i^k/\rho + T_i z^{k+1} + t_i) \\ &= d_i^k - \rho\,\mathbf{prox}_{\,g_i/\rho}(d_i^k/\rho)\ . \end{aligned}$$

Using Moreau identity (A.1) we have that

$$\begin{aligned}\lambda_i^{k+1} &= d_i^k - d_i^k + \mathbf{prox}_{\rho g_i^\star}(d_i^k) \\ &= \mathbf{prox}_{\rho g_i^\star}(d_i^k) \ ,\end{aligned}$$

as it directly follows from Property XXX in Appendix A.

In addition, step 2 of Algorithm 10 is the same as Step 3 of Algorithm 5. Subsequently, both algorithms take a proximal step in the direction of the negative gradient of the dual function, the only difference being the point at which the gradient is evaluated from the previous steps. Nothing changes when the accelerated versions of the algorithms are considered (FAMA and CPII), where, again, only the point of evaluation of the gradient differs. The aforementioned equivalence demonstrates the strong interconnections of the algorithms. Almost the same problem is tackled in different spaces, namely in the dual space for the Dual Proximal Gradient Method, in the primal space for AMA, where PDHG operates in both spaces, hence characterized as a primal-dual method.

References

[1] Hedy Attouch, Jérôme Bolte, Patrick Redont, and Antoine Soubeyran. Alternating proximal algorithms for weakly coupled convex minimization problems. Applications to dynamical games and PDE's. *J. Convex Anal.*, 2008.

[2] Ahmet BehÃğet AÃğikmese and Lars Blackmore. Lossless convexification of a class of optimal control problems with non-convex control constraints. *Automatica*, 2011.

[3] Heinz H Bauschke and Patrick L. Combettes. *Convex analysis and monotone operator theory in Hilbert spaces*. Springer Science+ Business Media, 2011.

[4] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Imaging Sci.*, 2009.

[5] Amir Beck and Marc Teboulle. A fast dual proximal gradient algorithm for convex minimization and applications. *Operations Research Letters*, 2014.

[6] Stephen Becker and Jalal Fadili. A quasi-Newton proximal splitting method. In *Advances in Neural Information Processing Systems 25*. 2012.

[7] Dimitri P. Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods (Optimization and Neural Computation Series)*. Athena Scientific, 1996.

[8] Dimitri P Bertsekas. *Nonlinear Programming*. Athena Scientific, 1999.

[9] Dimitri P. Bertsekas and John N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Prentice-Hall, Inc., 1989.

[10] Hai YANG Bing Sheng HE, Sheng Li WANG. A modified variable-penalty alternating directions method for monotone variational inequalities. *Journal of Computational Mathematics*, 2003.

[11] Radu Ioan Bot, Ernö Robert Csetnek, and Andre Heinrich. On the convergence rate improvement of a primal-dual splitting algorithm for solving monotone inclusion problems. *arXiv preprint arXiv:1303.2875*, 2013.

[12] S.P. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[13] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.*, 2011.

[14] A. Chambolle and T. Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of Mathematical Imaging and Vision*, 2011.

[15] Gong Chen and Marc Teboulle. A proximal-based decomposition method for convex minimization problems. *Math. Program.*, 1994.

[16] Eric Chu, Brendan OâĂŹDonoghue, Neal Parikh, and Stephen Boyd. A Primal-Dual Operator Splitting Method for Conic Optimization. 2013.

[17] Patrick L Combettes and Jean-Christophe Pesquet. Proximal splitting methods in signal processing. In *Fixed-point algorithms for inverse problems in science and engineering*, pages 185–212. Springer New York, 2011.

[18] Patrick L Combettes and Jean-Christophe Pesquet. Primal-dual splitting algorithm for solving inclusions with mixtures of composite, lipschitzian, and parallel-sum type monotone operators. *Set-Valued and variational analysis*, 20(2):307–330, 2012.

[19] Laurent Condat. A primal-dual splitting method for convex optimization involving Lipschitzian, proximable and linear composite terms. 2011.

[20] G. B. Dantzig and P. Wolfe. Decomposition Principle for Linear Programs. *Operations Research*, 1960.

[21] Wei Deng and Wotao Yin. On the global and linear convergence of the generalized alternating direction method of multipliers. *Rice CAAM technical report TR12-14*, 2012.

[22] Alexander Domahidi, Aldo U. Zgraggen, Melanie Nicole Zeilinger, Manfred Morari, and Colin Neil Jones. Efficient interior point methods for multistage problems arising in receding horizon control. In *CDC*, 2012.

[23] J. Douglas and H. H. Rachford. On the numerical solution of heat conduction problems in two and three space variables. *Comp. Math. Appl.*, 1956.

[24] Jonathan Eckstein and Dimitri P. Bertsekas. On the Douglas-Rachford Splitting Method and the Proximal Point Algorithm for Maximal Monotone Operators. *Mathematical Programming*, 1992.

[25] Ernie Esser, Xiaoqun Zhang, and Tony F. Chan. A general framework for a class of first order primal-dual algorithms for convex optimization in imaging science. *SIAM J. Img. Sci.*, 2010.

[26] J.E. Esser. *Primal Dual Algorithms for Convex Models and Applications to Image Restoration, Registration and Nonlocal Inpainting.* 2010.

[27] Hugh Everett. Generalized Lagrange Multiplier Method for Solving Problems of Optimum Allocation of Resources. *Operations Research*, 1963.

[28] J. Fadili. Variable metric monotone operator splitting. `http://www.math.u-bordeaux1.fr/~jaujol/conf2012/slide/Fadili.pdf`.

[29] H. J. Ferreau, H. G. Bock, and M. Diehl. An online active set strategy to overcome the limitations of explicit MPC. *International Journal of Robust and Nonlinear Control*, 2008.

[30] D. Gabay and B. Mercier. A dual algorithm for the solution of nonlinear variational problems via finite-element approximations. *Comp. Math. Appl.*, 1976.

[31] Euhanna Ghadimi, André Teixeira, Iman Shames, and Mikael Johansson. Optimal parameter selection for the alternating direction method of multipliers (ADMM): quadratic problems. *arXiv preprint arXiv:1306.2454*, 2013.

[32] Pontus Giselsson. Improving fast dual ascent for mpc - part ii: The embedded case. 2013.

[33] Pontus Giselsson and Stephen Boyd. Diagonal scaling in Douglas-Rachford splitting and ADMM. *53rd IEEE Conference on Decision and Control*, 2014.

[34] R. Glowinski and A. Marrocco. A Modification of the Arrow-Hurwicz Method for Search of Saddle Points. 1975.

[35] R Glowinski and Patrick Le Tallec. *Augmented Lagrangian And Operator-splitting Methods In Nonlinear Mechanics.* Society for Industrial and Applied Mathematics, 1989.

[36] A.J. Goldman and A.W. Tucker. Polyhedral convex cones. In *Linear Inequalities and Related Systems.* Princeton university Press, 1956.

[37] T. Goldstein, B. O'Donoghue, and S. Setzer. Fast Alternating Direction Optimization Methods. *arXiv.org*, 2012.

[38] Tom Goldstein, Ernie Esser, and Richard Baraniuk. Adaptive primal-dual hybrid gradient methods for saddle-point problems. *arXiv preprint arXiv:1305.0546*, 2013.

[39] Tom Goldstein and Stanley Osher. The split bregman method for l1-regularized problems. *SIAM J. Imaging Sciences*, 2009.

[40] Osman Güler. On the convergence of the proximal point algorithm for convex minimization. *SIAM J. Control Optim.*, 29(2):403–419, February 1991.

[41] E.N. Hartley, J.L. Jerez, A. Suardi, Jan M. Maciejowski, E.C. Kerrigan, and G. Constantinides. Predictive Control using an FPGA with Application to Aircraft Control. *IEEE Transactions on Control Systems Technology*, 2013.

[42] Bingsheng He and Xiaoming Yuan. Convergence Analysis of Primal-Dual Algorithms for a Saddle-Point Problem: From Contraction Perspective. *SIAM J. Imaging Sciences*, 2012.

[43] Bingsheng He and Xiaoming Yuan. On the $O(1/n)$ Convergence Rate of the Douglas-Rachford Alternating Direction Method. *SIAM J. Numerical Analysis*, 2012.

[44] B.S. He, H. Yang, and S.L. Wang. Alternating direction method with self-adaptive penalty parameters for monotone variational inequalities. *Journal of Optimization Theory and Applications*, 2000.

[45] R.M. Hestenes. Multiplier and gradient methods. *Journal of Optimization Theory and Applications*, 1969.

[46] L. Hurwicz K. J. Arrow and H. Uzawa. Studies in linear and non-linear programming. *Stanford University Press*, 1958.

[47] D. K. M. Kufoalor, S. Richter, L. Imsland, T. A. Johansen, M. Morari, and G. O. Eikrem. Embedded model predictive control on a plc using a primal-dual first-order method for a subsea separation process.

[48] T. Chan M. Zhu. An efficient primal-dual hybrid gradient algorithm for total variation image restoration. UCLA CAM Report, 2008.

[49] Jacob Mattingley and Stephen Boyd. CVXGEN: a code generator for embedded convex optimization. *Optimization and Engineering*, 2012.

[50] Zhi-Quan Luo Mingyi Hong. On the linear convergence of the alternating direction method of multipliers, 2012.

[51] J. J. Moreau. Fonctions convexes duales et points proximaux dans un espace hilbertien. *Comptes Rendus de l'Académie des Sciences (Paris), Série A*, 255, 1962.

[52] ArkadiÄŋ NemirovskiÄŋ and David Borisovich ïÿąIïÿăUdin. *Problem complexity and method efficiency in optimization*. Wiley-Interscience series in discrete mathematics. Wiley, Chichester, New York, 1983. A Wiley-Interscience publication.

[53] Yu. Nesterov. A method for solving the convex programming problem with convergence rate $o(1/k^2)$. *Dokl. Akad. Nouk SSSR*, 1983.

[54] Yu. Nesterov. *Introductory lectures on convex optimization: A basic course*. 2004.

[55] Jorge Nocedal and Stephen J Wright. *Numerical Optimization (2nd edition)*. Springer, 2006.

[56] B. O'Donoghue and E.J. Candes. Adaptive Restart for Accelerated Gradient Schemes. *arXiv.org*, 2012.

[57] B. O'Donoghue, G. Stathopoulos, and S. Boyd. A splitting method for optimal control. *IEEE Transactions on Control Systems Technology*, 2012.

[58] Brendan O'Donoghue, Eric Chu, Neal Parikh, and Stephen Boyd. Operator splitting for conic optimization via homogeneous self-dual embedding. *arXiv preprint arXiv:1312.3039*, 2013.

[59] Neal Parikh and Stephen Boyd. Proximal algorithms. *Foundations and Trends in Optimization*, 2014.

[60] B.T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics* , 4(5):1 – 17, 1964.

[61] L. Popov. A Modification of the Arrow-Hurwicz Method for Search of Saddle Points, journal = Mathematical Notes, year = 1980,.

[62] A.U. Raghunathan and S. Di Cairano. Optimal step-size selection in alternating direction method of multipliers for convex quadratic programs and model predictive control. In *International Symposium on Mathematical Theory of Networks and Systems (MTNS)*, pages 807–814, June 2014.

[63] S. Richter, C. Jones, and M. Morari. Certification Aspects of the Fast Gradient Method for Solving the Dual of Parametric Convex Programs. *Mathematical Methods of Operations Research*, 2013.

[64] R. Tyrrell Rockafellar. Monotone Operators and the Proximal Point Algorithm. *SIAM J. on Control and Optimization*, 1976.

[65] R.T. Rockafellar. Augmented lagrangians and applications of the proximal point algorithm in convex programming. *Mathematics of operations research*, 1956.

[66] Daniel Ruiz. A scaling algorithm to equilibrate both rows and columns norms in matrices. Technical report, 2001.

[67] Ron Shefi and Marc Teboulle. Rate of Convergence Analysis of Decomposition Methods Based on the Proximal Method of Multipliers for Convex Minimization. *SIAM Journal on Optimization*, 2014.

[68] N. Z. Shor, Krzysztof C. Kiwiel, and Andrzej Ruszcayǹski. *Minimization Methods for Non-differentiable Functions*. Springer-Verlag New York, Inc., 1985.

[69] P. Tseng. Applications of splitting algorithm to decomposition in convex programming and variational inequalities. *SIAM J. Control Optim.*, 1991.

[70] E. Ullmann. A Matlab toolbox for C-code generation for first order methods. Master's thesis, ETH Zurich, 2011.

[71] Băng Công Vũ. A splitting algorithm for dual monotone inclusions involving cocoercive operators. *Adv. Comput. Math.*, 38(3):667–681, 2013.

[72] M. Burger X. Zhang and S. Osher. A Unified Primal-Dual Algorithm Framework Based on Bregman Iteration. *Journal of Scientific Computing*, 2011.

[73] Xiaojie Xu, Pi fang Hung, and Yinyu Ye. A simplified homogeneous and self-dual linear programming algorithm and its implementation. *Annals of Operations Research*, 1996.

[74] Yinyu Ye, Michael J Todd, and Shinji Mizuno. An $o(\sqrt{nL})$-iteration homogeneous and self-dual linear programming algorithm. *Mathematics of Operations Research*, 1994.

[75] M. Zhu and T. Chan. An efficient primal-dual hybrid gradient algorithm for total variation image restoration. *UCLA CAM Report*, 2008.